D.S.P. of circuit design for P.W.M. D/A conversion.

Hiorns, R E

D S.P. I circuit design for PW M. D/A. co version

Digital Signal Processing and Circuit Design for Pulse Width Modulating Digital to Analogue Converters

R.E. Hiorns

Department of Electrical and Electronic Engineering
King's College
University of London
1994

This thesis is submitted in part fulfilment of the requirements for the degree of 'Doctor of Philosophy'.

to Sarah

Abstract

This thesis describes techniques for implementing a digital to analogue (D/A) converter which produces output signal power sufficient for audio applications.

Digital pulse width modulation (DPWM) is attractive for highly efficient power D/A conversion because it can be used with a class D output configuration. Using DPWM causes various theoretical problems in satisfying the linearity and resolution requirements of an audio power amplifier. These difficulties are discussed for cases with and without feedback. The emulation of a high linearity analogue pulse width modulator is proposed, and the performance of one implementation is compared with systems using alternative linearising techniques from the literature.

The practical problems of constructing a power D/A converter using DPWM are also considered, and in particular how to avoid excessive circuit clock speed by using error spectrum shaping (ESS) techniques. New filter objectives and design techniques are proposed to ameliorate the problems arising from using ESS in combination with DPWM. The design technique itself is examined as a difficult optimisation problem. A theoretical solution for the optimal filter shape and novel ESS architectures are also presented.

Circuit designs for the entire system are discussed with emphasis on solving difficulties introduced by high speed, resolution and linearity. Examples are presented demonstrating how the difficulties can be overcome. The system performance is evaluated by comparing measurements taken from hardware with those from computer simulation and theory.

A commercial prototype has been developed using the work in this thesis and it demonstrates that amplifiers based on DPWM can compete with the currently available alternatives.

Table of Contents

List of Illustrations

List of Tables

Acknowledgements

Numerous people have helped me on my way to a better understanding of DSP, PWM and implementation limitations through technical discussion, constructing hardware and in some cases allowing me to use their hardware or software for my own research. From this group I should like to make a special mention of Dr. Jason Goldberg who is responsible for developing the algorithm now known as Pseudo Natural PWM, and who gave his own time generously for many theoretical discussions. I should also like to thank Allan Paul who provided invaluable proof reading and let me use 'Simulate', a suite of DSP software developed from Jason Goldberg's programmes; this helped me to complete accurate simulation of DPWMs as well as filter design, sample rate conversion, open loop pre-compensation and spectral estimation.

I should also like to thank by name, Robert Bowman, Howard Farrar and Kevin Davis. Rob was responsible for implementing the noise shaper design discussed in chapter 6 as an ASIC as well as designing and building a PCB to support the 96002 DSP on which I was able to test weighted average pre-compensation. Howard also designed and built a PCB, this time to house a 56004 DSP on which noise shaping and enhanced sampling were implemented. Kevin laid out the PCB implementation of PWM 4, as described in chapter 6, as well as helping with the assembly of stereo interface circuitry (also as in chapter 6).

I should like to thank my supervisor Dr. M.B. Sandler for giving me the opportunity to study and the Science and Engineering Research Council, the British Technology Group, and B&W (Loudspeakers) who provided financial support for this work. Also, I should like to thank members of the department of Electronic and Electrical Engineering of King's College for valuable discussions and seminars on signal processing and circuit design and in particular Dr. Mike Waters and Dr. Mike Page-Jones.

Lastly, I should like to thank my fiancee and parents for literally years of support, advice, encouragement and patience, without which this research would never have been completed.

Chapter 1 : Introduction

1.1 : Overview of the Chapter

In recent years, there has been a gradual transition from the use of analogue storage, processing and reproduction of signals to digital versions, performing equivalent tasks. The digital versions use techniques that are less prone to noise and distortion and yield more predictable performance. The increased noise immunity allows more sophisticated signal conditioning to be applied in digital signal processing (DSP), while keeping system noise in check.

Conversion from the digital form back to an analogue signal is required whenever digitally encoded data is to be used in the 'real world' but conventional digital to analogue converters (DACs) produce small signals that need subsequent amplification. This amplification stage can spoil the advantages gained in the digital storage or processing, so new ways to produce large analogue signals directly from the digital version are needed. Such a system will be referred to as a 'Digital Amplifier'.

DACs are employed in many applications, audio reproduction being a particularly demanding one. Consequently, audio reproduction will be used to quantify design requirements for a high quality digital amplifier in terms of resolution, linearity, bandwidth, efficiency and output power level.

Most currently available DACs not only produce a small analogue signal, but also distort and add noise to it. Furthermore, fabrication of high quality DACs using traditional topologies pushes construction techniques to their limits, making them expensive to produce and restricting further development. A cheaper system with greater potential for development, the ability to produce a large output and which still satisfies audio application requirements, is the subject of this thesis.

The proposed solution to this problem is based on pulse width modulation (PWM). Although power amplifiers based on this principle have been constructed before, they have all been analogue in nature, prone to noise and distortion and unable to accept digital input. This proposal makes use of novel implementations of a digital pulse width modulator (DPWM). The accuracy of these circuits is derived from the stability of a quartz crystal oscillator used to define the clock in the DPWM unit, quartz oscillators being one of the most stable references known to mankind.

DPWMs capable of the resolution and bandwidth required for audio applications require a counting speed of near 3 GHz, which is extremely difficult to achieve. To avoid this problem, DSP is used to reduce the pulse width resolution requirements with minimal noise compromise. This process is commonly called 'Noise shaping' and is central to the practicality of using DPWM. Novel filters and structures for implementing this stage of DSP will be presented.

All PWM techniques produce distortion to some degree, and the DPWM output requires filtering before use to remove high frequency components from its spectrum. By using more DSP to increase the sample rate of the system ('Oversampling' by interpolation), the output filtering can be simplified and the PWM distortion is reduced. Unfortunately the distorted output is still unsuitable for audio use, but a third stage of DSP can be included to pre-distort the signal and cancel the output's harmonic content. Use of this novel approach and its implementation will be compared to alternative DPWM strategies and structures, and will be seen to be crucial to satisfying the linearity requirements.

Lastly, a power efficient, switched, output stage is proposed to complete the digital amplifier.

Shown below is a system diagram to summarise the proposed circuit and signal processing elements.

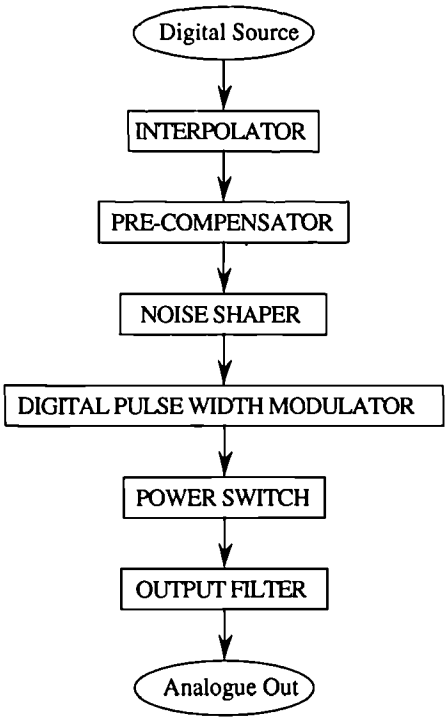


Figure 1.1.1.a : Basic Block Diagram of a System proposed for a Digital Amplifier.

Several problems exist in the DSP part of this system since the noise shaper and DPWM interact and any signal or noise added after the pre-compensator is not properly compensated for. Similarly, there are circuit difficulties associated with using a power switch (speed, accuracy and hysteresis) or an output filter operating at high power (saturation and heating). Reducing radio frequency interference is also difficult in this kind of output stage. Lastly, problems arise between the DSP and circuits since the DSP is designed assuming the output switch operates from an ideal voltage source (constant voltage rails) which may not be the case without a special power supply design. These problems will be discussed later in the thesis.

In this chapter, existing DAC and amplifier techniques will be examined, looking in particular at their performance and shortfalls. The proposed digital amplifier will be looked at on a similar basis. The requirements of audio reproduction will then be presented along with techniques for measuring whether or not those requirements have been met. These issues will then be summarised in preparation for detailed scrutiny of each of the blocks shown above, chapter by chapter.

1.2 : Existing DAC Techniques and their Limitations.

1.2.1 : D/A Converter Categories

Several techniques exist for implementing D/A conversion and these will be discussed briefly before examining the drawbacks of each. The techniques can be split into five approaches :

1) Averaging DACs,
2) DACs based on scaled resistor networks (or on an 'R-2R ladder' network),
3) DACs using scaled current sources (often called 'current switching' DACs),
4) Oversampled $\Sigma\Delta$ DACs (sold under the trade name 'bitstream')
5) Combined Noise Shaping - PWM based DACs (marketed as 'MASH' and 'PEM').

Averaging DACs operate by using frequency to voltage (F/V) conversion or pulse width modulation (PWM). F/V versions use an input of frequency proportional to the digital input word from which a train of pulses is created, each with an equal quantum of charge. PWM based versions use a train of pulses with fixed 'off-time' between but each is given a width proportional to the digital signal value. Either train of pulses can be low pass filtered or integrated to obtain a moving average of the pulsed signal whose voltage follows the digital input. Some output ripple results, so to keep the amplitude of this ripple less than the quantisation noise in an 'N' bit system using the F/V version with minimum input period 'T_0', the time constant, 'T', of a simple R-C filter, should satisfy :

$$T \geq \mathrm{Log}_e(2).(N+1).T_0 \qquad \text{Equation 1.2.1.a}$$

This sets the settling time for a 10 bit DAC using a maximum frequency of 100 kHz in the F/V stage to about 0.6 ms. PWM versions don't require quite such a long settling time, but both versions are only suited to gradually varying signals such as large motors or low frequency switched mode power supplies (SMPS). One elegant application is temperature control where the filtering can ignored (relying on thermal 'inertia') and burst firing can be used (switching integer AC supply cycles to the load).

DACs based on scaled resistor networks are particularly popular since these can give near instantaneous output. Fast response allows these DACs to be used in the feedback of successive approximation analogue to digital converters (ADCs), doubling the applications this technique is used for. The basic principle of operation relies on the virtual earth principle in operational amplifiers with large negative feedback and an earthed non-inverting input. Numerous voltage sources can be summed at the inverting input (the virtual earth) after passing through a resistor to define its weighting. The negative feedback and high gain combine to maintain the summing node at zero volts and thus prevent interaction between the sources. If these voltages are scaled using a set of binary weighted resistors the voltage sources can be switched on or off dependent on the binary digits of the input word and the summed voltage is proportional to the value represented by the input. Digital codes using non-binary weighting (eg. A-law as defined by CCITT) can also be accommodated by using corresponding weighting in the resistor set. A schematic diagram of the essential components required for a binary version this type of converter is shown below :

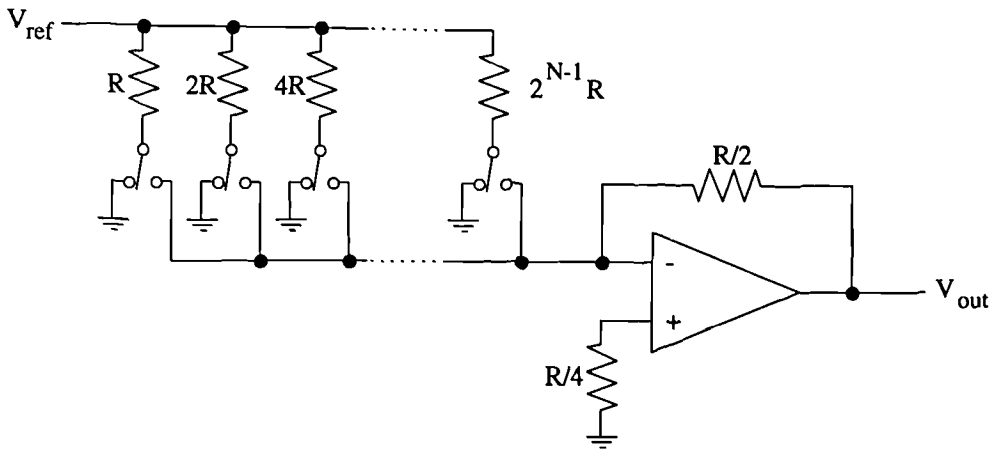


Figure 1.2.1.a : Basic Scaled Resistor Digital to Analogue Converter Schematic

It should be noted that large current variation is required of the reference voltage unless the circuits pass the same current regardless of the state of the bits controlling them. To achieve this, the idle circuits are connected to ground, and because the virtual earth is very close to ground, the current variation is kept small [HOR80].

Where high dynamic range is required (ie. a large number of bits need to be represented) the above circuit becomes unacceptable. This is because even though resistance has the largest range of all fundamental quantities, matching values to the accuracy of the smallest over more than four decades range in value is virtually impossible. To overcome this problem, a modified circuit has been developed based on a ladder of resistors with values 'R' and '2R' as shown below :

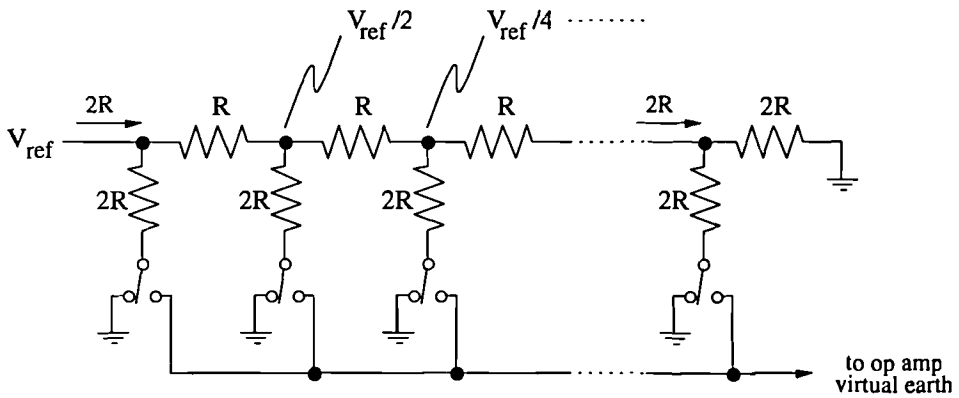


Figure 1.2.1.b : R-2R Resistor Ladder Digital to Analogue Converter Schematic

At each tapping point in the ladder, the voltage halves, but the resistance of the remaining ladder sections down to earth (to the right) remains equal to 2R. As before, switching idle circuit sections to earth maintains constant current operation of the voltage reference reducing the variations in demand put on the reference. Using the above circuit, the range problems in large wordlength DACs can be solved, but accuracy in either of these circuits is limited by the stability of the reference and the matching of the components. Furthermore, the linearity, speed of operation and drift with temperature are limited by the operational amplifier used [SED82].

Implementing a DAC with a summation of binary weighted current sources allows faster operation and avoids most of the temperature drift problems of the last circuits. Unfortunately, this also implies that the output current is proportional to the digital input, rather then the output voltage.

The binary weighted currents can be defined by an R-2R ladder as used before and each constant current sink can be used in the common emitter of an npn differential amplifier circuit. By drawing current from earth rather than from the current output the differential amplifiers can be used to control the total current pulled from the output. On a similar basis, *pnp* transistors could be used in the differential amplifier with current *sources* to *push* current *into* the output. A circuit segment of this sort of circuit is shown below :

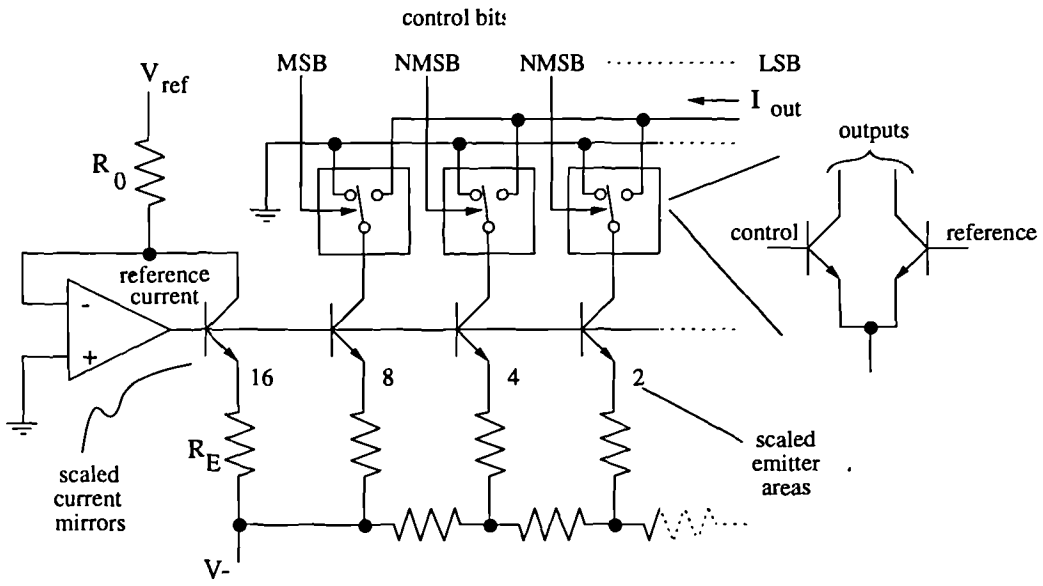


Figure 1.2.1.c : Basic Current Summing DAC Circuit Schematic.

For extremely fast operation the differential amplifiers can be replaced by Schottky diode based switches. Settling times of 25 ns. can be achieved using this technique although the circuit then suffers from increased noise and reduced output compliance.

Both of the current summing DACs can be followed by an operational amplifier in the transimpedance configuration to create an output voltage or for small voltage swings, a simple resistor. Unfortunately, the current summing DAC accuracy is still dependent on the R-2R ladder accuracy since it is this which defines the current ratios. In integrated circuit versions, laser trimming of the 'on-chip' resistors after production can ensure high performance but this process is very expensive.

Another DAC structure uses a single bit $\Sigma\Delta$ modulator to produce a single bit output ('bitstream') from a highly oversampled multi-bit input. Combining this technique with digital interpolation, a one-bit D/A and an output low pass filter produces very high quality output which can be repeated reliably once integrated. The output prior to filtering is essentially pulse density modulated with a sample rate some 32 to 128 times oversampled compared to the input (typically 1-5 MHz) so output filtering can be kept simple maintaining good phase and amplitude linearity at the output. Unfortunately, the high frequency of the output prevents this technique being used at high power.

Lastly, multistage noise shaping can be used to provide lower noise versions of the ΣΔ modulator DAC and the overflow caused by using multiple stages can be pulse width modulated to avoid saturated output. So called 'MASH' converters are a development of the 'bitstream' techniques already mentioned. A variant on this theme has been called pulse edge modulation ('PEM'). This relies on *multi-bit* noise shaping to pre-empt overflow, using large input oversampling, and a pulse width modulated output. Both these techniques use such a high frequency pulse repetition rate in the PWM stage that high power implementations are not feasible but it is the use of similar ideas to these that the proposed digital amplifier is based on.

1.2.2 : D/A Converter Problems

Ideally, the output from a DAC should be proportional to its digital input; deviation from this arises from several causes and can be characterised for comparison. Three main areas cause problems : speed, noise and linearity.

As discussed before, averaging DACs are slow and seldom used except in coarse control systems. ΣΔ and combined noise shaper - PWM DACs can handle considerably faster changing signals and resistor ladder and current summing DACs are the fastest. Traditionally, the speed of a DAC has been defined by its settling time but in the ΣΔ or noise shaping cases the bandwidth is a more appropriate measure since high frequency content in the output of such DACs implies that they never 'settle'. In these converters the usable bandwidth is traded off against available resolution but a signal to noise ratio (SNR) of about 100 dB over 1.5% of the input is quite possible.

Noise in any DAC can be generated by noisy components, but more considerable problems arise in separating the digital and analogue parts of the converter. The appearance of digital clocking artifacts in the output is common (sometimes called 'clock feedthrough') and can only be avoided by careful design and layout.

Control signals in the DAC can contribute to the output level so that despite correct operation of a ladder or scaling system, the output levels are not quite evenly spaced. This effect, and poor component matching, lead to linearity errors and limit the resolution that can be achieved; five types are of particular interest : offset error, gain error, missing codes, non-monotonicity and differential linearity error (DLE). Transfer functions illustrating these effects are shown below :

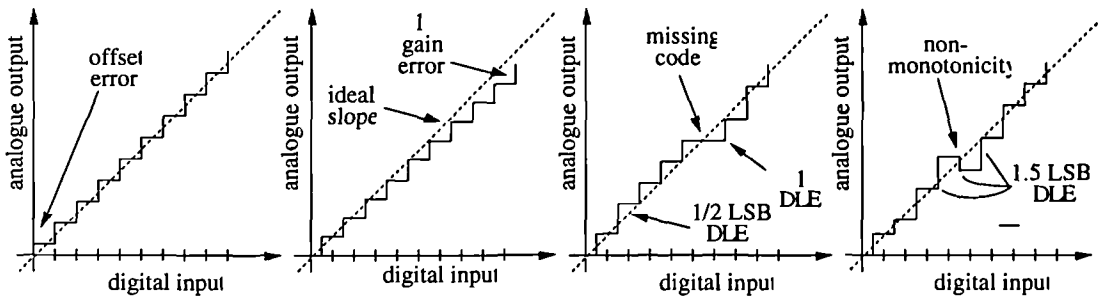


Figure 1.2.2.a : Linearity Errors Commonly found in DACs

Many DACs incorporate trimming to reduce these effects both in production and construction stages.

All DACs require some time to settle since internal switching cannot be guaranteed to occur simultaneously and overshoot in the step response is often permitted to achieve a fast response time. To avoid either sources of amplitude error contributing to the output, a sample and hold circuit can be used after the DAC. Such circuits are often called 'de-glitching' circuits and are clocked just prior to new data appearing from the DAC itself. This re-sampling inherently has a response that falls off with frequency depending on the sampling window duration ('aperture error'). For a window duration, t_0, a sampling period, t_s , and an input frequency, ω_{sig}, the frequency response is :

$$H|e^{j\omega}| = \frac{t_0}{t_s} . Sinc(\frac{1}{2}\omega_{sig} t_0)$$

Equation 1.2.2.a

Choosing a small value of t_0 limits the signal to noise ratio because only a small amount of energy is passed in a short time, but choosing a large value of t_0 increases the amount of settling and overshoot error being considered to be signal. A compromise of $t_0/t_s \approx 1/4$ is commonly used leading to the half sample rate amplitude being suppressed by 3%.

The quantised analogue output from the de-glitcher changes at the clock rate of the DAC so further filtering is required to recover the baseband signal alone and suppress spectral replicates. So called 'recovery filters' can be designed but the required filter order is large unless the sample rate is considerably higher than twice the signal bandwidth (which is the minimum or 'Nyquist' rate). Such oversampling is commonly included before the DAC to alleviate this filter design and then higher cutoff frequencies in the recovery filter can be used permitting better phase response in the signal band. Special designs of oversampling filter or recovery filter can also be used to compensate for the in-band signal roll off introduced by the de-glitcher.

Oversampled $\Sigma\Delta$ modulators or DACs incorporating noise shaping already make use of high sample rates for other reasons so these converters already have the advantages described above.

1.3 : Existing Power Amplification Techniques and their Limitations.

1.3.1 : Power Amplifier Classes

Many circuits exist for implementing power amplifiers but only some of these have found applications in broadband power amplification. The efficiency and capabilities of these circuits is largely dependent on the proportion of current that is used for biasing, lost in switching, or passed to the load. For this reason power amplifiers are classified according to the conducted current duty cycle for a sinusoidal input. A list of the commonly used classifications [PAR92] is given below :

	Class	Characteristic(s)
1)	A	linear operation : 360° conduction,
2)	B	'push-pull' operation : 180° conduction,
3)	AB	class B with bias for slight conduction cycle overlap (to avoid crossover distortion),
4)	C	< 180° conduction (usually ≈120°), resonant loading is assumed,
5)	D	conduction only near zero or saturation (< 5%),
6)	E	class D with a capacitor to move the conduction cycle phase for smaller I_{on} ,
7)	F	class C with a load tuned to the 3 rd. harmonic of the input frequency,
8)	S	non-resonant class D using PWM and very fast switching.

Classes 'AB' and 'S' are subsets of the basic set, popularised by audio signal power amplification.

Classes 'C', 'E' and 'F' are not usually considered suitable for broadband amplification since these all depend on a resonant load. This is particularly unfortunate since class 'C' can provide a typical power efficiency of 85 % and classes 'E' and 'F' approach 100%. In this introduction the other classes will be concentrated on with example circuits before examining their shortfalls in performance.

Class A amplification is the least efficient class with a theoretical best of only 50% (which fortunately coincides with maximum signal amplitude). This low efficiency results in heating of power devices so that such amplifiers require significant heat-sinking and are only suitable for modest power levels. Whatever heat is produced often alters circuit characteristics so thermal considerations need to be included in potential designs. Usually circuits use biasing to move the operating point on the transistor's input-output transfer characteristics to the the middle of the linear range, and add to this a signal to be amplified. Using the linear part of the transistor's characteristics provides low distortion, but the biasing wastes considerable amounts of power, requiring larger output devices and a larger power supply for the same capabilities as other classes can provide. A basic circuit is shown below :

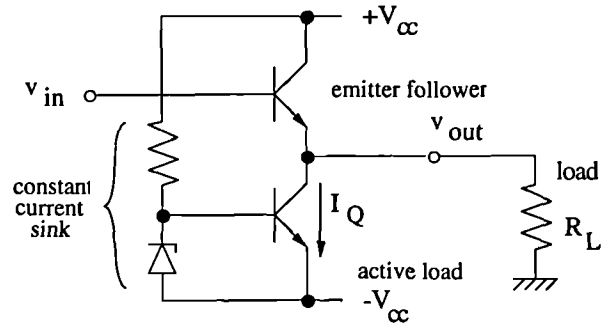


Figure 1.3.1.a : Basic Schematic for a Class A Amplifier Topology

To overcome the inefficiencies of class A, the two transistors can be used in a push-pull fashion if complimentary transistors are used. By using one transistor to handle positive and the other to handle negative input voltages, the mid-point biasing can be avoided allowing theoretical efficiency figures approaching 80%. The transistor which is not in use is cut off so no power is wasted in this device for half of the cycle, substantially reducing the heating within it an hence allowing less heat-sinking to be used. The wasted power is substantially reduced, and the output power per device can be increased, leading to an overall reduction of a factor of 4.9 in the size of power supply used. A typical circuit for this is shown below :

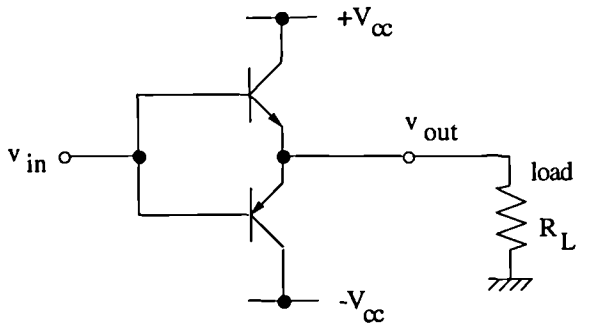
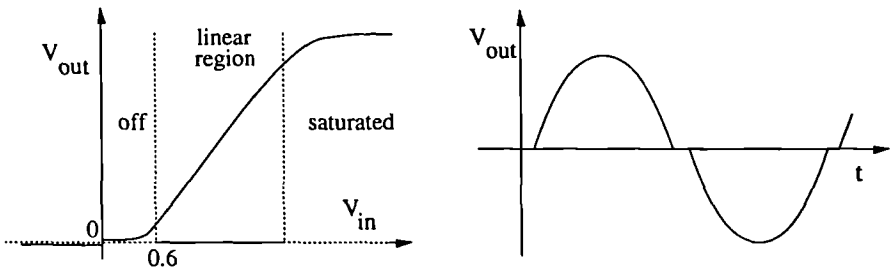


Figure 1.3.1.b : Basic Schematic for a Class B Amplifier Topology

Unfortunately, the transfer characteristics of most commonly used bipolar junction transistors are only linear over a restricted range which does not include zero. As the transfer function approaches zero, the transistor begins to switch off prematurely so each in half cycle conduction stops in a 'dead band' around zero. This leads to distortion of the output wave as it crosses zero (commonly called 'crossover distortion') as shown below:



Transfer Characteristic Crossover Distortion

Figure 1.3.1.c : Transistor Transfer Function & Resulting Crossover Distortion in Class B Amplifiers

The biasing of each transistor to avoid it working at zero current is often included by the addition of two diodes with voltage drop similar to that in the base emitter junction of the power transistors. Such a circuit is shown below; it allows conduction over slightly more than 180° of the cycle and is commonly designated Class AB.

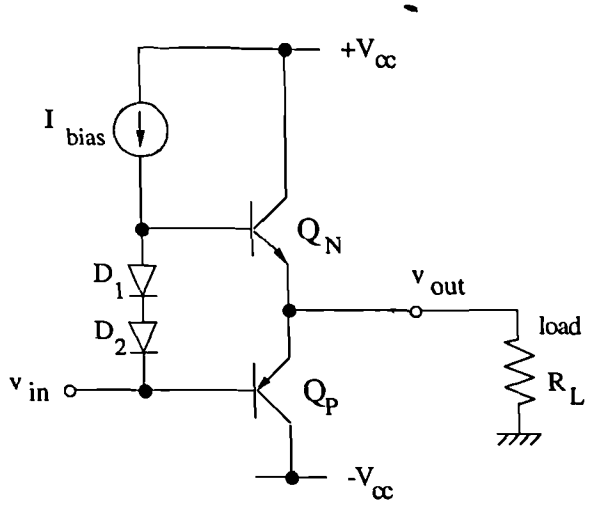


Figure 1.3.1.d : Basic Schematic for a Class AB Amplifier Topology

Class D amplifiers work on a totally different principle, whereby current and voltage are never intended to co-exist in the output transistors. In this way the power loss in the amplifier is very small, but to accomplish this the amplifier is not operated in a linear fashion at all. An example circuit is shown below along with the current voltage waveforms over one cycle.

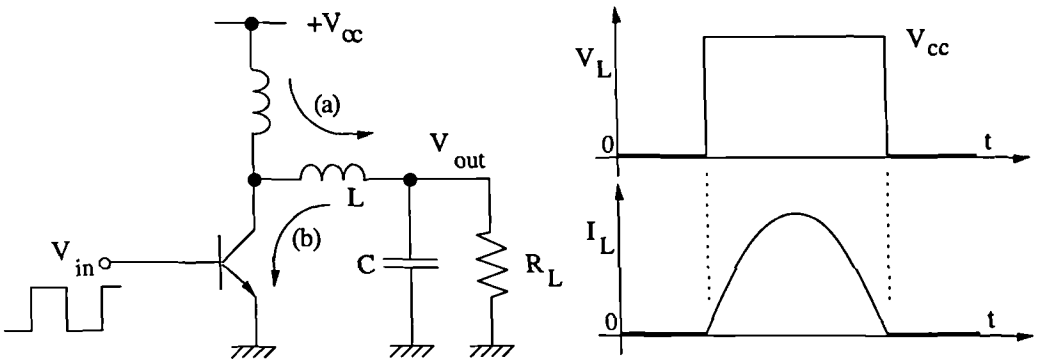


Figure 1.3.1.e : Basic Schematic for a Resonant Class D Amplifier Topology

By switching between two levels extremely quickly very little power is lost through current and voltage being present in the transistor at the same time. Using a series L-C resonant load, the transistor can be timed to switch on when the supply current has fallen to near zero (b). Once saturated, the device voltage drop is small so little power is wasted. As the load current rises the transistor voltage falls to a point at which it equals zero and the transistor can be switched off (a). Again, little power is lost in the device but this time through the lack of voltage drop across the device. Once the transistor is switched off the falling supply current is forced into the load until the next cycle begins.

A variant on this circuit scheme is to provide complimentary switching in the top half of the circuit which is also operated either saturated or off. This allows the resonant load to be replaced with an arbitrary load and a broadband response can be achieved. If the complimentary inputs are pulse width modulated and the output voltage is low pass filtered, a viable audio amplifier can be constructed (sometimes called Class 'S') as shown below :

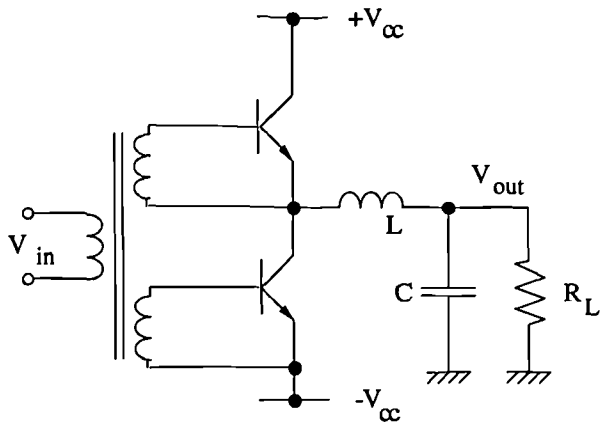


Figure 1.3.1.f ; Revised Schematic for a Broadband Class D Amplifier Topology

Although the switching process is highly non-linear, the spectral performance inherent to the pulse width modulation used can be achieved at a high power level and switching efficiencies approaching 100% can theoretically be achieved. Spectral components associated with the PWM carrier and sidebands force the overall efficiency to be reduced so figures nearer 95% are more common when calculated as the *useful* output signal power over the consumed power.

High switching efficiency can allow small switching devices to be used to control large output powers and still avoid large heat-sinks. Furthermore, efficient use of the power supply enables smaller power supplies to be used for the same output power. However, the pulse width modulation produces some distortion and requires the output voltage levels to be very close to ideal, requiring additional regulation in audio applications. Furthermore, high frequency, high current pulses at conduction overlap can destroy the output devices if not carefully controlled.

1.3.2 : Power Amplifier Problems

As already touched on, amplifiers suffer from heating problems that make suitable design essential for the chosen output devices. The different topologies vary in efficiency, and as already seen, those which offer higher efficiency suffer less from the limiting and potentially destructive effects of heating. Thus, these topologies will require smaller heat-sinking and can achieve the required power output with a smaller power supply.

Power supply stability is a further problem, since any limit on the available current prevents the output devices from controlling the load current in proportion to the input current. This eventually leads to clipping and general odd order harmonic distortion. For audio use with Class A, B or AB, bipolar junction transistors (BJT) are commonly used with the advantage that these devices control the output current variation and it is this which is responsible for the movement of the loudspeaker coils. MOSFET devices are now becoming more popular because their negative temperature coefficient can prevent damage in cases of overheating but these are not yet preferred for the best quality amplifiers. Class D amplifiers operate by controlling the output voltage which in turn drives the required load current; because of this, they are less tolerant of power supply variation. On a similar basis, load impedance variation (with frequency or different loads) causes more problems in Class D circuits.

1.4 : The All Digital Power Amplifier.

Having examined some of the shortfalls of the currently available DACs and power amplifier topologies the proposed digital amplifier structure will be examined. Six main areas will be looked at as listed below, the connection of which is shown in figure 1.1.1.a .

1) Digital pulse width modulator (& crystal oscillator),
2) Noise shaper (& dither source),
3) Interpolator,
4) Pre-compensator,
5) Power switch (& power supply),
6) output filter.

1.4.1 : Functions of the Basic Elements

The heart of the conversion process proposed is pulse width modulation (PWM). This is essentially a D/A conversion stage but since a totally digital implementation of this can be constructed (DPWM) there is very little added noise incurred and only simple filtering required after it to recover the analogue signal. A basic block diagram of how such a circuit converts digital input into trailing edged modulation is shown below :

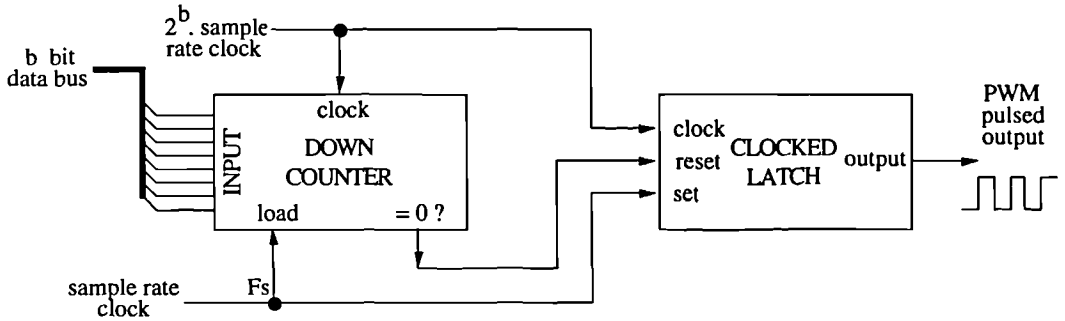


Figure 1.4.1.a : Block Diagram of a Digital, Trailing Edged, Pulse Width Modulator

DPWM can produce a two state (ie. digital) output which can be subsequently 'amplified' by using the small digital signal to switch a larger version. By keeping the signal digital, the signal can be 'cleaned-up' by re-clocking from an independent supply; this reduces the opportunity for noise to enter the signal until it is at a high power level where the added noise is insignificant.

DPWMs use counters to define the edge times of the output pulses and it is the accuracy of these which puts an absolute limit on the resolution the system can achieve. The accuracy of the counters depends on the accuracy of the clock driving them, so a particularly stable and noise free clock is required to achieve good performance. Quartz crystal based clocks are ideal for this purpose, offering good long term stability and low phase noise.

The counting speed required in a DPWM is at least the product of the number of output widths each pulse can be given and the pulse repetition rate. This leads to difficulties because for high resolution systems the counting speed becomes enormous. For instance, audio signal data is currently stored on compact discs using 16 bit resolution and 44.1 kHz sample rate, implying 65536 output levels in a period of 22.6 μs. This would require a counter operating at 2.89 GHz which is expensive if not impossible to achieve. Simple recovery filtering requires an even higher sample rate so a drastic reduction in the number of bits has to be used to alleviate the counting speed difficulties.

Noise shaping is a technique for reducing the number of bits required to express a signal which allows a portion of the spectrum to be retained with whatever resolution the input provides. In this process the less significant bits of the signal are forced to zero (adding wideband quantisation noise), so these do not need to be modulated; the error so introduced is retrospectively corrected for by adding a filtered version of the error to the next sample (spectrally shaped to lie outside the band of interest). The information capacity of the output is greatly reduced when compared to the input, but by oversampling the signal, sufficient capacity can be provided at the output to represent the signal. For instance, 16 bit quality over $1/8^{th}$ of the input band can be represented within an 8 bit signal using only a fifth order filter. The spectrally shaped error can then be placed in the spare part (ie. 7/8 ^{ths}) of the spectrum. Although oversampling increases the required DPWM counting speed, it enables enormous savings in speed to be achieved through lower wordlength. A structure for implementing a noise shaper is shown below with the shaping filter marked '$H(z^{-1})$' :

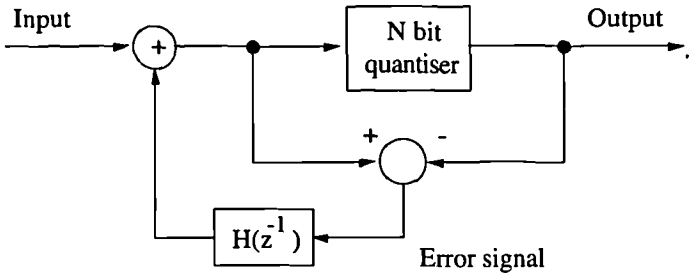


Figure 1.4.1.b : Basic Noise Shaper Structure

The recursive filtering used in a noise shaper can produce distortion for small signals and idle channel spuriae. These are particularly undesirable in audio systems since the ear is particularly sensitive in the absence of large signals. To destroy the structure of these effects a small noise source can be added which is called 'dither'. Although the signal to noise ratio is reduced slightly by doing this, the subjective performance is enhanced (VAN89). Dither can be generated by numerous means but one of the simplest is a pseudo-random noise generator based on a maximal length sequence from a shift register with feedback. This signal is then added into the signal path inside the noise shaper as will be shown in chapter 4.

Having designed a DPWM and noise shaper with compatible parameters, the oversampling ratio will be known and standard DSP techniques can be used to produce an interpolator to digitally emulate oversampling (see chapter 3). It should be noted that oversampling and interpolation are not quite the same thing, since low rate sampling concentrates quantisation noise in the baseband. Once low rate sampling has been performed, its quantisation noise contribution cannot be removed, whatever the sample rate, whereas an oversampled signal has its quantisation noise evenly spread over all bands.

PWM whether implemented digitally or in continuous time (analogue input) produces distortion. Various modulation algorithms exist (chapter 2) and the best of these should be chosen as a compromise between the complexity of the algorithm required and the characteristics of the distortion it produces. Some algorithms can be implemented directly in hardware as a part of the modulator, but more complicated algorithms that perhaps operate over limited bandwidth have to be placed before the noise shaper as indicated in figure 1.1.1.a . These algorithms will be called 'pre-compensation' since they alter the form of the output pulse so as to compensate for the distortion the DPWM would otherwise produce.

With the above components, a signal level version of the interpolated, noise shaped, pulse width modulated signal will be achieved (with the chosen modulation type as conjured up by the compensator + DPWM combination). If this signal were to be filtered, a low power DAC could be constructed which could be used before class AB amplification or suchlike. By taking the digital signal before filtering and applying it to power MOSFET switches, a high power digital signal can be created which after filtering at the high power level, yields a large analogue output.

The power switch stage is only possible in this type of output because the switching frequency is reasonably low (ie. hundreds of kilohertz not megahertz). Heating in the output devices is the main barrier to higher frequency pulse repetition rates being used which is why alternative pulsed output DACs cannot be used as a digital amplifier in a similar way (ie. 'ΣΔ', 'MASH' or 'PEM').

The power switch is operated in saturated or off modes and hence is class D in nature, with the rail separation controlling the output power level. Adjustable power rails can be used as a convenient way of producing a 'volume' control while maintaining resolution as the level changes but this does complicate the supply and switch design.

The output filter is essentially a recovery filter, but owing to the the oversampling required for the noise shaping used, this is usually a simple passive design using low order and providing good signal band phase linearity.

Although the output stages are highly efficient (based on a class D principle) and a simple power supply would be expected, the power supply has to be tightly controlled with good compliance so that the switched voltage levels applied to the filter are constant. This is best achieved by using a switched mode power supply (SMPS) which compliments the high power efficiency achieved in the power switch itself.

1.4.2 : Difficulties within the Digital Amplifier

Although the noise shaper - pulse width modulator - class D switch system is proposed as a good DAC and amplification method, it is not without its problems. Clock compatibility, noise shaper design, pre-compensator design and power stage problems all have to be overcome to make the system work. These are addressed in detail in the following chapters but a brief observation of the more significant ones are mentioned here.

It should be remembered that for high quality to be preserved, the edge time *accuracy* must be at least that required without noise shaping even though the edge time *resolution* can be reduced. Using a local crystal oscillator is the best way of ensuring low phase noise and hence accurately timed pulse edges but if built into the digital amplifier, this would require that other connected units use the same

clock (or division of it) to maintain synchronous data transfer. Frequency multiplication of data coming into the digital amplifier can be used as an alternative clock source but the stability and phase noise of such a source are considerably worse than direct from an oscillator. Asynchronous sample rate conversion (ASRC) can be used to permit re-sampling of the signal stream [ADA92] but this introduces jitter of up to ±150 ps. [ADI93]. ASRC operates by interpolating the input stream by a large factor, and outputting the closest sample to the desired sampling instant; signal storage is then employed to allow the sampling ratio to vary slowly in time. Although this is a viable solution to clock incompatibility (allowing pulse edge jitter to be *locally* controlled), it is not yet known what the audible effects of such sample rate conversion is, so in situations where the source clock can be operated in a 'slave' mode this will be preferred.

The use of oversampling and noise shaping is essential to allow realistic counting speeds in the DPWM and does help to make the PWM process appear more linear, but this combination is not problem free. Unfortunately, the shaped noise which lies outside the audio band can be modulated to recur in-band depending on the design of the noise shaping filter, the number of bits 'dropped', the oversampling ratio and the modulation type used. This limits the minimum oversampling ratio that can be used and demands complicated design procedures for the noise shaping filter (see chapter 4).

Noise shaper - DPWM systems rely on counters operated at speeds well below that implied by the input signal resolution (eg. c.100 MHz); the modulation types directly available to such a system are in fact those called 'uniformly sampled' which create significant distortion (chapter 2). Alternative modulation types exist with lower distortion and different distortion distributions with frequency. Pre-compensation can be added to construct a signal after modulation that produces less distortion. Routines for this vary in complexity but those proposed here (chapter 5) cannot be operated properly after noise shaping since the out of band noise inhibits the compensation calculation technique. By placing the pre-compensation before the noise shaper, only the signal is compensated for and the shaped noise will be distorted; this distortion is not necessarily a concern since this will be removed by the recovery filter anyway. Fortunately, the high frequency performance of the uniformly sampled PWM is superior to that from the alternatives so this ordering produces good results although the modulated shaped noise sets a lower limit on the resolution that can be expected from this type of system.

One favourable modulation type uses successive samples to dictate the width of left and right portions of each pulse, with each being measured from a regular timing marker. This is of particular interest because under certain conditions the output can be arranged to produce only odd order distortion terms (see section 2.3.6). Since lower index distortion terms tend to be the most significant in PWM distortion characteristics this is particularly useful. This modulation type does assume that alternate samples are converted to width vectors with opposite polarity which means that when noise shaping is used in conjunction with this modulation type, successive samples add to cancel some of the noise shaping effects. This problem can be solved by separating adjacent samples dependence after noise shaping, and a special noise shaper characteristic is proposed for this in chapter 4.

Between the DPWM and the output switch itself, pre-drivers are required for most power MOSFETs. Hysteresis and delay in either the pre-drivers or the power MOSFETS can introduce substantial distortion. Duty cycles of less than 10 % or greater than 90 % are also difficult to represent properly since switching transients interfere with each other. To limit the modulation depth 'guard

bands' can be used in the PWM and these prevent the use of over-large or small pulse widths. Problems also arise in getting perfect complimentary switching action; any overlap allows a current spike to pass from one rail directly to the other rather then to or from the load which can destroy the output devices. To reduce this special filtering has been developed (see chapter 7).

Power MOSFETs are not ideal switches in two particular respects. Firstly, the gate has some capacitance which limits the turn on and turn off delay according to the impedance of the driver and secondly, the channel resistance of a MOSFET is non-zero so a voltage drop between the supply and the filter is bound to occur. Also, these devices have a parasitic anti-parallel diode which has slow recovery characteristics allowing continued conduction after turn-off unless specially designed types are used (eg. FREDFETs). On top of all this, MOSFETs are limited in the $\partial I/\partial t$ and $\partial V/\partial t$ that they can handle without further parasitic effects coming into operation. These problems force the output stages to be carefully designed and limit the overall performance that can be achieved from a class D output although, as will be shown, performance can be as good as currently available amplifiers can achieve.

Care is also required to avoid saturation of the inductors used after a class D output design since the output filter may be required to pass several amps of current leading to high flux densities in iron or ferrite cores. This can be avoided by using air cored inductors although this implies an increased amplifier size. High power ferrites as developed for use in DC-DC converters and power supplies may be a viable alternative.

Finally, although SMPS units provide highly efficient regulation of the voltage rails, their chopping frequency often appears as output ripple (even though it should be filtered out). To avoid 'beating' with the switching frequency of the DPWM itself, the oscillator used to define the chopping frequency inside the SMPS should be locked to a division of the PWM clock.

1.5 : Requirements and System Assessment.

1.5.1 : Audio Applications & their Requirements

There are numerous areas that can benefit from the use of digital amplification both for audio reproduction and other amplifier applications. Audio amplification will be used as an example with which the requirements can be defined for three contrasting applications :

1) Public address,
2) Portable amplifiers (eg. personal stereos),
3) Hi-Fidelity.

These applications vary in three ways: output power level, acceptable distortion level, and efficiency. Public address requires particularly high output power levels with moderate distortion and efficiency. Personal stereo requires only low power levels, considerable distortion and noise can tolerated but exceptional efficiency is required. Hi-fidelity applications require moderate power levels, as little distortion as possible and efficiency is not of great concern.

Users' opinions of system requirements vary considerably. To avoid entering any discussion of the merits of one method of testing versus another, the amplifier requirements will be copied from the performance of existing examples of 'top of the range' Hi-Fi. Although these are not definitive margins between acceptable and unacceptable performance, the requirements do serve as a guide. To avoid direct comparison of one manufacturer with another, an example of a DAC, a preamplifier and a power amplifier from the same manufacturer are presented below (similar performance has been observed form alternative manufacturers).

Device name	Audiolab 8000DAC
DAC technique	Phillips 20 bit PDM ("bitstream")
SNR achieved	> 100 dB (A weighted)
THD+N achieved	< 0.003%, 20-20 KHz, FS o/p
frequency response	±0.5 dB 2-20 KHz
oversampling	8x Interpolator (BB), PRR = 384x

Device name	Audiolab 8000C (pre-amp)
SNR achieved	92 dB (IHF A @ 0.5V o/p)
TH+IMD achieved	< 0.01%, 20-20 KHz
frequency response	±0.5 dB 20-20 KHz, -3 dB 2-65KHz

Device name	Audiolab 8000P (power amplifier)
amplifier class	100 W, Class AB
SNR achieved	> 95 dB (IHF A @ 0 dBW)
THD achieved	< 0.05%, 20-20 KHz, any level o/p
frequency response	±0.3 dB 20-20 KHz, -3 dB 3-75 KHz
output capability	4-16 Ω load, 30V, 40 A capacity
overall efficiency	44% at full load
weight	9.0 kilogrammes

Figure 1.5.1.a : Three Tables Of Typical Audio Systems' Performance

1.5.2 : Comparison Techniques

Four performance parameters are of particular concern in assessing an audio system. These are the system resolution, the system linearity, the system frequency response and the system power efficiency. Frequency response can easily be measured using a swept tone and measuring output amplitude; power efficiency is simply the ratio of output signal power over the DC input power. Resolution and linearity are expressed indirectly from three measurements : signal to noise ratio (SNR), total harmonic distortion (THD) and intermodulation distortion (IMD).

SNR is the ratio between the total noise power and a signal of specified power; the point of measurement should be stated since loading and grounding conditions can alter the proportion of unwanted noise that is included. This figure is usually quoted in decibels and can be calculated as :

$$SNR = 10.Log_{10}\left(\frac{\text{SIGNAL POWER}}{\text{NOISE POWER}}\right) \qquad \text{Equation 1.5.2.a}$$

In many cases the test signal is sinusoidal of known amplitude and the r.m.s. noise level is known in which case the SNR can be calculated directly as :

$$SNR = 20.Log_{10}\left(\frac{\text{SIGNAL AMPLITUDE}}{\sqrt{2}.\text{RMS NOISE AMPLITUDE}}\right) \qquad \text{Equation 1.5.2.b}$$

When phase distortion makes subtraction of the scaled input from the output difficult or in cases of particularly high dynamic range when measurement error is large compared to the noise amplitude, spectral separation can be used to pick out the signal from the noise. By taking a FFT of the temporal signal (after appropriate band-limiting, sampling, quantisation and windowing) the spectrum can be found and from this approximation of the power spectral density, components of the noise can be summed up. It should be noted that windowing allows some spectral leakage so that power in the frequency bins adjacent to the signal misrepresent the spectral components that actually lie there. To overcome this error in the measurement, a bias can be set up indicating the number of bins considered to belong to the signal so that known cases (such as a pure tone) satisfy the theoretical SNR for that type of signal, sample rate, window, quantisation, etc. This method is used in the assessment of both simulations and prototypes of the proposed digital amplifier since it should have a large SNR.

A careful choice of window has to be made since most windows introduce low level spectral leakage across the whole band as well as broadening of the central lobe. While a bias figure can compensate for the errors introduced by the broadening of the dominant spectral lobe, and defining 'exclusion' bands can compensate for SNR errors introduced by other known signals (eg. in twin tone tests), the broadband spectral leakage can only be kept small by using windows with monotonic decay of side lobes *and* high order FFTs (typically, sixteen thousand points or more).

Five typical windows are drawn below to show the contrast in spectral leakage that can be incurred by windowing [HAR78,NUT81]. From this it should be noted that the rectangular and hamming windows have a particularly narrow central lobe, but a high level of broadband leakage, making them useful only for tests where central lobe width is of concern. The 3 and 4 term optimised windows provide a good compromise between central lobe width and side lobe levels and are used in graphical plots for visual inspection using small FFT orders (<2000). Their side lobes decay at only 6 dB per octave (from -75 dB and -98.5 dB at 20 frequency bins deviation respectively) and so would interfere with measurement of signals with dynamic range above 100 dB.

The exact 4 term Nuttall window has high side lobes (-61 dB) but 42 dB/octave decay rate and is preferred for most measurements. It has a moderate central lobe width of +/- 4 frequency bins so appropriately high order FFTs are used with this to shrink the bandwidth affected by leakage.

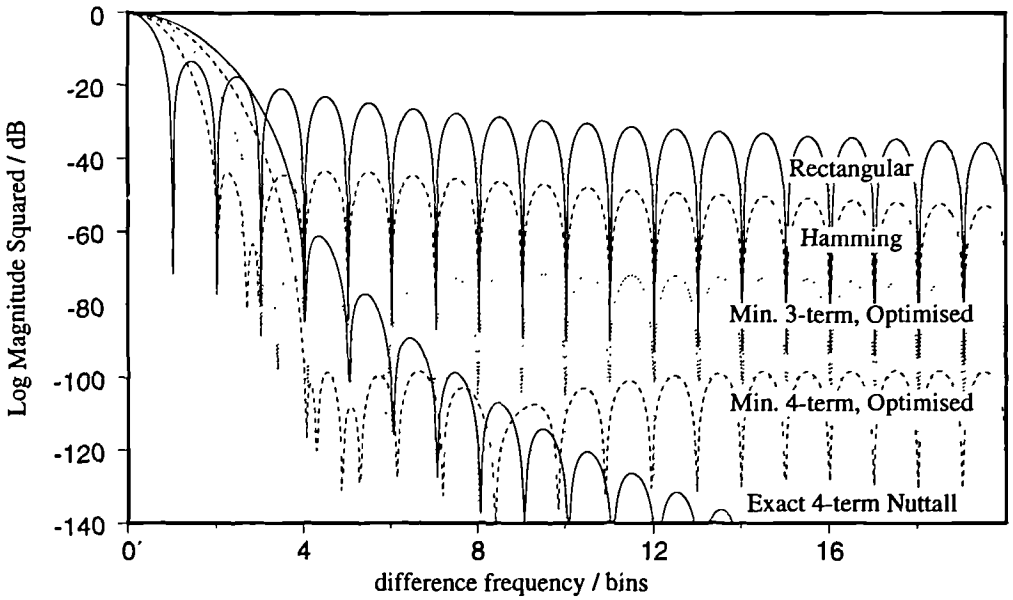


Figure 1.5.2.a : Five Windows of Interest, Showing Near-Signal Spectral Leakage.

Distortion measurements are usually expressed in percent and figures of less that 0.5 % are considered suitable for audio reproduction (PAR92). If measurement is performed in the analogue domain harmonic and non-harmonic components are often grouped together yielding a total distortion figure (from which the separate components cannot be deduced). Measurements taken in the frequency domain are usually somewhat easier to identify since intermodulation appears at sum and difference frequencies of those present in the input. Simple tests based on twin tones are particularly useful since relatively few intermodulation products appear in this way and these can then be distinguished from the harmonically related distortion.

Conventional SNR, THD and IMD tests are taken over all frequencies but in the case of pulsed output systems such as PWM or $\Sigma\Delta$ based DACs, this misrepresents the audible performance of the system by including high frequency carrier terms. To produce measurements that more closely predict audible performance, a band-limited version of the tests can be performed. If the measurements are made in the frequency domain this is easy since terms lying outside the audio band can be ignored.

Conventional measurement of SNR, THD and IMD give equal weighting to all frequency components which is not representative of perceived quality when the system is listened to. The ear's dynamics are more sensitive in the 300 Hz to 3000 Hz band and errors made at these frequencies are far more significant than those at say 15 KHz. To overcome the shortfalls of the conventional measurements, the spectrum can be weighted to compliment the ear's constant loudness curve (as a function of frequency). Shown below are the constant loudness curves for some example levels †.

† The number of phons is numerically equivalent to the number of decibels above the inaudible sound pressure level of a 1 KHz tone, 0.00002 Pa (= 0 dB). Average listening levels ≈ 40 phon (40 dB SPL), a whisper ≈ 20 phon, street noise ≈ 60 phon and a jet aircraft ≈ 120 phon.

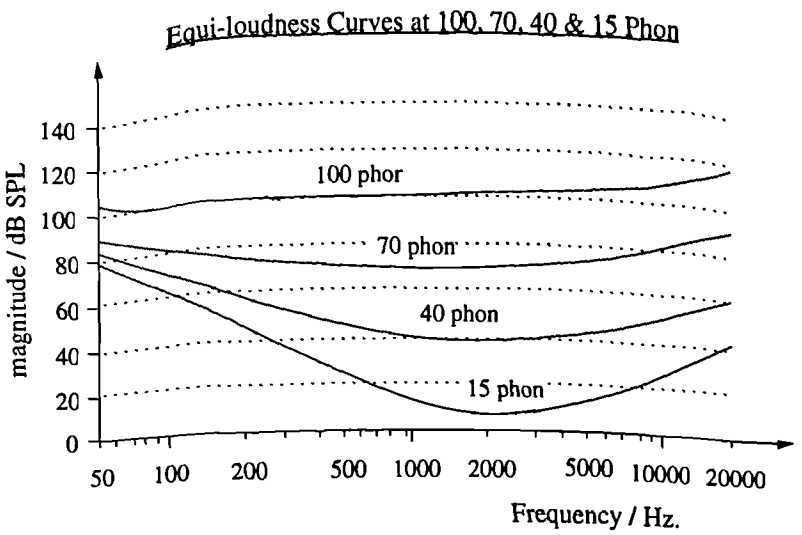


Figure 1.5.2.b : Constant Loudness Curves for the Human Ear [PAR92]

After scaling for unity power gain overall, the compliment of the average listening curve (40 phon) is commonly known as the A-weighting curve; from the low listening level curve (15 phon) the E-weighting curve is derived. Perceived SNR in the presence of a signal of average listening level can be calculated by A-weighting the spectrum, and perceived SNR for very quiet signals can be calculated by E-weighting the spectrum. The A-weighting and a simplified description for the E-weighting called the 'modified E-weighting' [VAN89] are shown below.

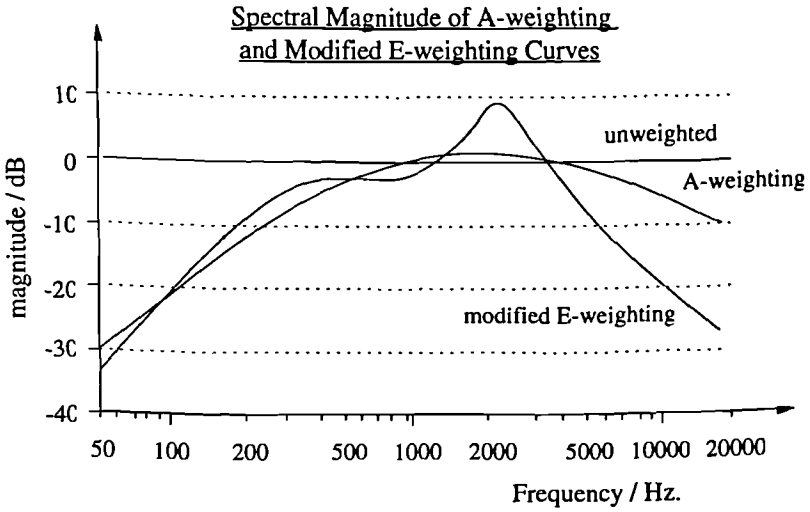


Figure 1.5.2.c : Modified E-Weighting Definition and Frequency Response

The perceived signal when listening to several tones becomes confused if one is substantially larger than another. In effect, the larger signal masks other frequencies so that the perceived signal quality is higher than the SNR or THD may suggest. To gain an understanding of the relative importance of spuriae in the spectrum, the spectral masking curves for a second tone in the presence of a 1.2 KHz tone at various listening levels are plotted below [WEG24]. From these, the adjacent frequencies can be seen to be quite insensitive to spuriae, and higher frequencies can be seen to be more masked than lower frequencies.

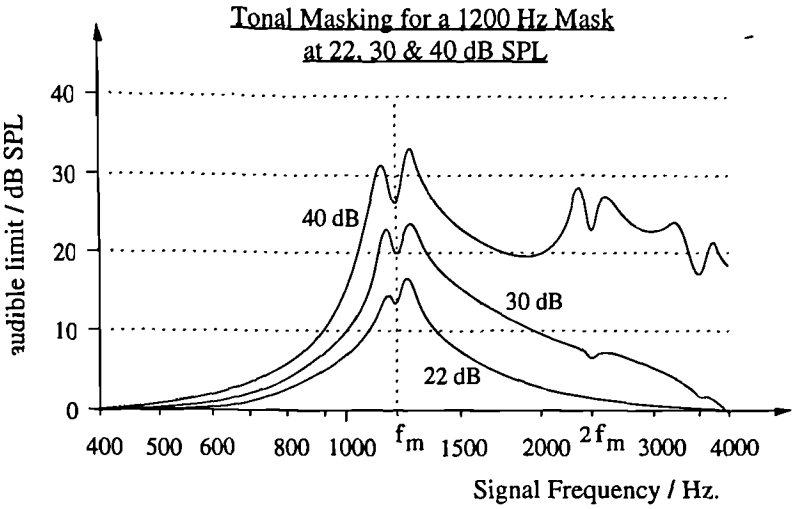


Figure 1.5.2.d : Spectral Masking From Various Amplitude Tones.

In the same way as spectral masking occurs by large amplitude signals, temporal masking occurs. Before and after a loud sound the ear does not distinguish sound as well as normally, allowing an amplifier with a less than ideal transient response to be acceptable [YOS77]. Shown below is the degree to which this masking covers the times near the transient. The axes are not marked since the amplitude and duration which is masked are a function of the amplitude and duration of the 'burst' of sound used as the masking function.

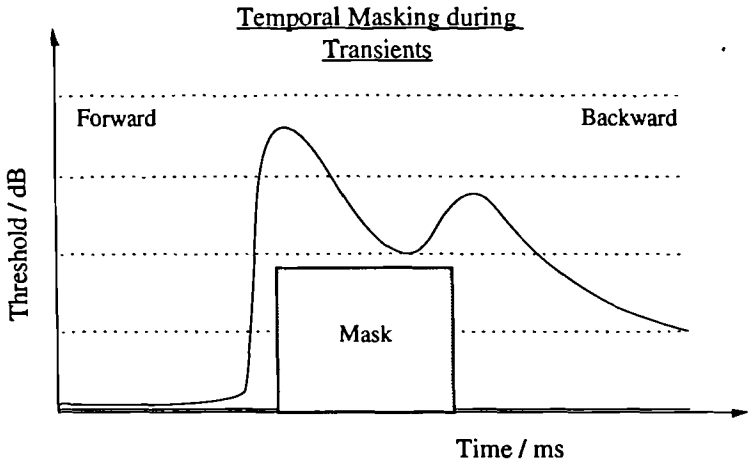


Figure 1.5.2.e : The Characteristic of Temporal Masking From a Burst of Sound.

Such temporal masking can cover transient terms that are visible as IMD in spectral analysis, so care has to be taken in interpreting IMD test results before an amplifier's IMD is considered unacceptable.

Large amplitude signals can introduce problems in measurement as well. Where a high dynamic range is to be measured using a lower dynamic range test-set, the imperfections recorded are mostly from the measurement system, not the system under test. Under special conditions this can be avoided; for instance, where a tone is used as the input signal, the output can be notch filtered to suppress the known signal. The remaining bands can then be amplified and analysed by FFT without danger of overloading the input stages (typically, an A/D converter). This technique is used repeatedly in chapters 6 and 7 where large tones are used to highlight imperfections in PWM based hardware.

1.6 : Summary

Various digital to analogue converter and power amplifier topologies exist but all rely on a low power analogue interface which is prone to noise. To avoid this a 'digital amplifier' is proposed, using a combination of these two functional blocks without analogue interconnections. A potential solution could be based on digital pulse width modulation, noise shaping and a class D amplifier.

Digital to analogue converters use time averaged pulsed output techniques or operate by using scaled resistor networks to define currents or voltages that can be summed. Large bandwidth can be achieved from the scaled resistor techniques but matching in the resistor network is considered a drawback since high resolution and linearity require difficult component matching.

Highly oversampled pulsed output techniques based on low wordlength pulse width modulation (PWM) or $\Sigma\Delta$ modulation provide less dependency on component matching, but produce pulses at such a high frequency that digital amplification cannot be applied.

Power amplifiers can be classified into several groups with class AB being the most popular for its reasonable power efficiency and linearity. Class A alternatives are considerably less efficient and suffer heat dissipation problems at high output power levels. Class D is very power efficient but not linear due to its switching action (this operates in either 'saturated' or 'off' modes). The use of class D in combination with PWM (sometimes called class S) is considered a foundation to the proposed digital amplifier although analogue PWMs are not considered useful because of their poor noise tolerance. Power amplifiers suffer from non-linearity, noise, limited bandwidth and low efficiency. These are considered the measures by which any proposed amplifier can be compared with existing systems.

The proposed digital amplifier consists of six main blocks : digital PWM to effect D/A conversion, oversampled noise shaping to reduce clock speeds in the PWM (to make construction feasible), interpolation to increase the sample rate (simulating oversampling), pre-compensation to give a more linear overall transfer function, a class D power switch to increase the amplitude of the digital signal, and low pass filtering to suppress switching components at high frequency (a recovery filter).

Edge accuracy is essential so quartz based clocking is proposed to provide this. An internally derived clock of this kind requires synchronisation with external digital data sources. Phase locked loop frequency multiplication is suggested when source devices cannot be 'slaved' as required, but this requires the use of asynchronous sample rate conversion to allow re-sampling of the input under the control of a local oscillator. Power supply regulation is considered essential and a switched mode power supply (SMPS) is suggested for this. This should also be synchronised to the amplifier to avoid 'beating' between the amplifier output and the SMPS unit. The output of high power class D stages produce high frequency components at high power levels. Filtering, shielding and good circuit design are required to sufficiently reduce radio frequency emission.

Existing audio amplifier performance is proposed as a 'yardstick' for the required performance of a digital amplifier. A summary of these is : SNR > 95 dB (A weighted), THD < 0.05 %, flat frequency response to within $\pm$0.5 dB (20 Hz - 20 kHz) and a power output between 10 and 100W. Techniques for assessing the performance of amplifiers also include allowances for psychoacoustic effects such as masking and weighting which may alleviate the requirements detailed above.

Chapter 2 : PWM Characteristics

2.1.1 : Introduction

Pulse width modulation is a term used to cover many variations on the theme of controlling the duty cycle of a fixed frequency pulsed output. In analogue electronics, this conversion of the input signal into the pulsed signal can be achieved by using a comparison to a sampling signal, typically a sawtooth or triangle wave (yielding 'naturally sampled PWM'). Where the input is higher than the comparison signal the output is switched on and where lower, off.

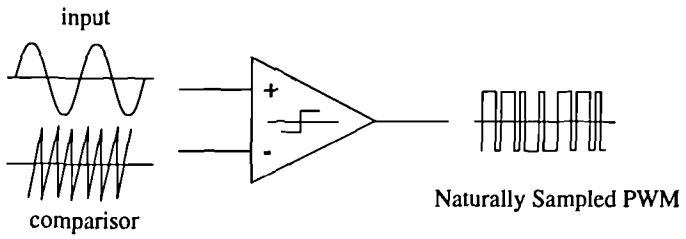


Figure 2.1.1.a : A Conceptual Schematic Diagram of an Analogue Pulse Width Modulator

Three techniques for modifying the output spectrum characteristics are particularly useful :

(1) changing the sawtooth wave gradient or frequency,

(2) changing the shape of the comparison signal, and

(3) pre-sampling the input signal.

Increasing the gradient of the sawtooth reduces the possible duty cycle range (the modulation depth, 'M') while increasing the frequency of the sawtooth increases the output pulse frequency (the carrier frequency, 'ω_c', or pulse repetition rate, 'PRR') :

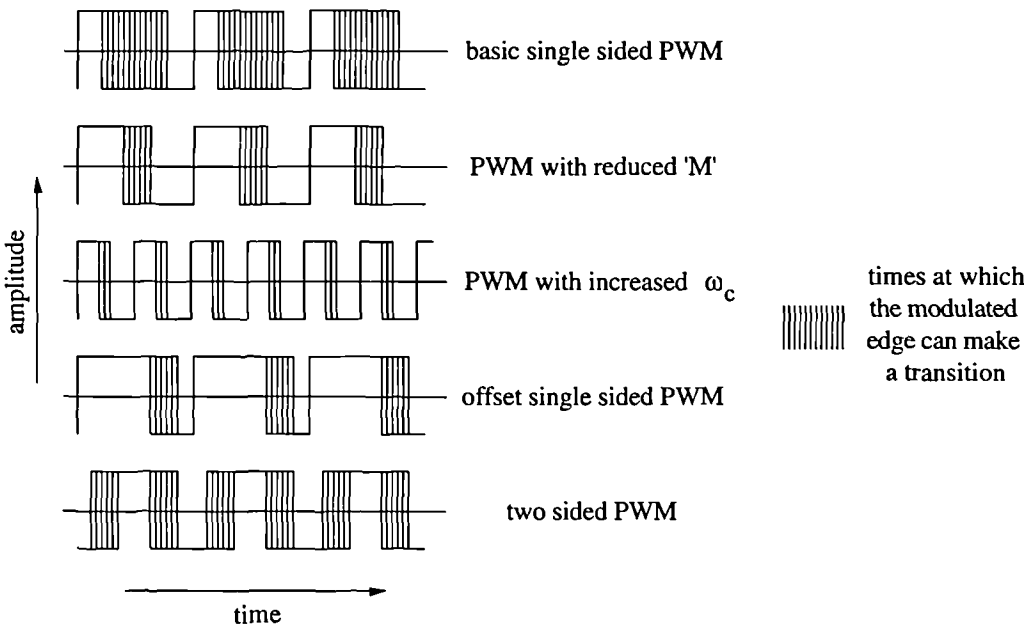


Figure 2.1.1.b : PWM Signals Arising From Altered Comparison Signals

Alterations to the comparison signal shape can create various modulation types; three are particularly well known :

(1) single sided modulation (sawtooth comparison signal),

(2) two edged modulation (triangular comparison signal), and

(3) modulation with a width offset to the pulses (from a DC offset between the input and comparison signal).

Using a sample and hold circuit on the input signal prior to comparison forces regular sampling. So called 'uniformly sampled PWM' has far inferior spectral performance to the 'naturally sampled PWM' described above as will be demonstrated shortly.

Most pulse width modulators provide a two level output, although a three level version can be achieved by matching the polarity of the output pulse to the input signal and controlling the width of the pulse by a rectified version of the input signal. Switched outputs (class D) using two levels have been designated 'Class AD' and those using three levels, 'Class BD' [MAR70]. Class 'BD' will not be discussed further since high linearity is required in audio applications and is easier to achieve in a two level output (where amplitude matching of the positive and negative levels can be avoided).

Comparators which are fast enough to handle a large input signal bandwidth (several hundreds of kilohertz) are not available with high resolution and undistorted performance (SNR > 95 dB and THD < 0.05%), so digital pulse width modulators (DPWMs) have recently been constructed to overcome these problems [HIO91a]. These operate by setting an output latch and employing a clocked counter, which can be loaded with a duration value, to reset the output on count-completion. By using a digital input (the duration value), quantisation noise is introduced into the input signal; by using a clocked output, the pulse widths are quantised to coincide with the clock. Both these quantisation effects alter the spectral performance of the DPWM compared to the analogue versions; for high resolution test tones the differences have been found to be small [HIO90b, HIO91a]. Analogue derivations of spectral performance will be used in this chapter to simplify the comparison of the various modulation techniques for single tones. For more complicated signals computer simulation will be used.

In this chapter the various classifications will be discussed (section 2.2) and the theoretical tonal performance will be presented and compared for the simple modulation cases (section 2.3). In section 2.4, distortion trends with modulation index and oversampling ratio will be examined and the level of harmonic, intermodulation and sideband distortion will be compared between modulation types. Section 2.5 summarises the effects which determine the choices of circuits and signal processing algorithms in later chapters.

2.2 : PWM Classifications

2.2.1 : Sampling Type Classifications

PWM in its continuous-time form, can be produced by comparison of the modulating signal with a sampling signal (whose shape and frequency will define the spectral performance of the output). The most common sampling (or comparison) waveform is a sawtooth, which produces pulse widths proportional to the signal amplitude at the time of the modulated edge. This process which varies the sampling instant with the signal amplitude is called natural sampling. A diagram of this for the case of trailing edge pulse width modulation (TEPWM) is shown below :

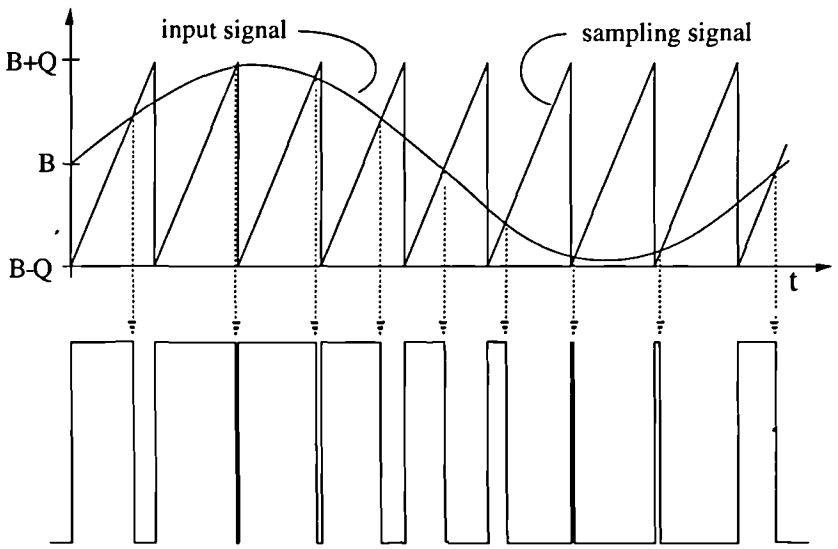


Figure 2.2.1.a : Naturally Sampled, TEPWM

It can be seen from above that the signal varies during the pulse interval, so the time of the modulated edge can only be guaranteed to be found (as a unique solution) if the rate of change of the signal is smaller than the rate of change of the sawtooth; ie. for a sinusoidal input, S, of amplitude Q, frequency ω_v and DC offset B, and a sampling signal of similar amplitude but frequency ω_c :

$$\frac{\partial S}{\partial t} < \frac{Q.\omega_c}{2.\pi} \qquad \text{Equation 2.2.1.a}$$

$$\Rightarrow \quad \omega_v < \frac{\omega_c}{2.\pi} \qquad \text{Equation 2.2.1.b}$$

Clearly this is a tighter constraint than the Nyquist sampling rate where only a factor of two is required between the signal and sampling frequencies.

In contrast to natural sampling as shown above, the input signal can be sampled at regular intervals, representing the value of the input at the unmodulated edge. Conversion of the sampled amplitudes to pulse durations can still be accomplished by comparison to a sawtooth wave but in this case the signal must be held constant over the sampling period. A 'sample and hold' circuit can be used to do this if placed in the signal path before the comparator and triggered at the unmodulated edge. The revised waveforms at the conversion stage are shown overleaf.

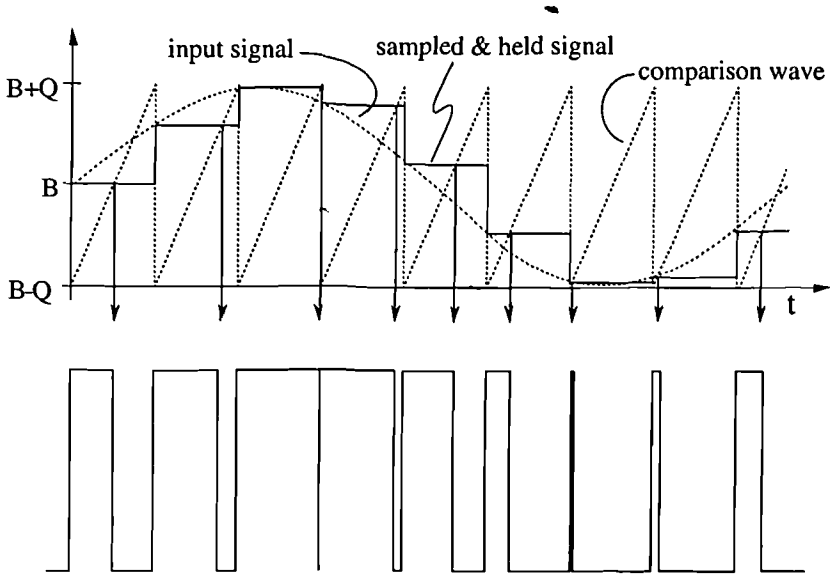


Figure 2.2.1.b : Sampling Waveforms at the Comparator Input Yielding Uniformly Sampled TEPWM

By comparing the sampling instants in the two sampling systems, the natural sampling scheme can be seen to produce longer pulses than the uniformly sampled for signals with positive derivatives (with respect to time) and shorter than the uniform for negative derivatives. In general terms, the natural sampler could be modelled by a discrete-time system taking uniformly sampled input with derivative terms in its transfer function, anticipating the changes in the signal.

Both schemes produce pulses which if modelled as plates cut from a sheet with mass could be said to have centres of mass which are not regularly spaced, but vary with the pulse width. This analogy can be taken one step further, by introducing the concept of energy application centres which like the centres of mass, are not regularly spaced in time. Using this concept, it is simple to see that for the uniformly sampled PWM, the energy which should have been applied at the sampling instants has been delayed by an amount dependent on the signal. ie.: the process is varying in time and hence not 'LTI' (linear, time-invariant). This will be confirmed later in the chapter by observing harmonic distortion in the tone spectra of uniformly sampled PWM signals. It is also important to note that the mapping of input sample amplitudes to output pulse widths is one-to-one. From this it can be deduced that despite the non-LTI performance, the information within the input signal can be recovered uniquely from the output signal. Thus a pre-distorted input signal can be constructed to avoid distortion in the output provided equation 2.2.1.b has been satisfied and the non-uniform sampling introduced by mapping amplitude vectors into time vectors does not introduce significant overlapping frequency sidebands.

Curiously, the natural sampling technique's 'anticipation' introduces distortion that cancels the harmonic distortion of the moving energy application centres (this will be shown more rigorously in section 2.3.2) leaving only sideband distortion terms. This feature makes natural sampling particularly attractive for linear conversion at low frequencies, assuming that subsequent filtering can be used to remove the sideband terms.

Digital systems commonly provide regularly sampled data so special techniques have to be used to take advantage of natural sampling's lower harmonic distortion. These will be returned to in chapter 5.

2.2.2 : Basic Edge Position Algorithms using One Sample per Pulse

With either natural sampling or uniform sampling schemes, the pulse can be constructed to modulate the leading edge (LEPWM) or the trailing edge (TEPWM). Simple time reversal of the sawtooth used for the trailing edged modulation used above as an example yields the appropriate waveform for LEPWM as shown below :

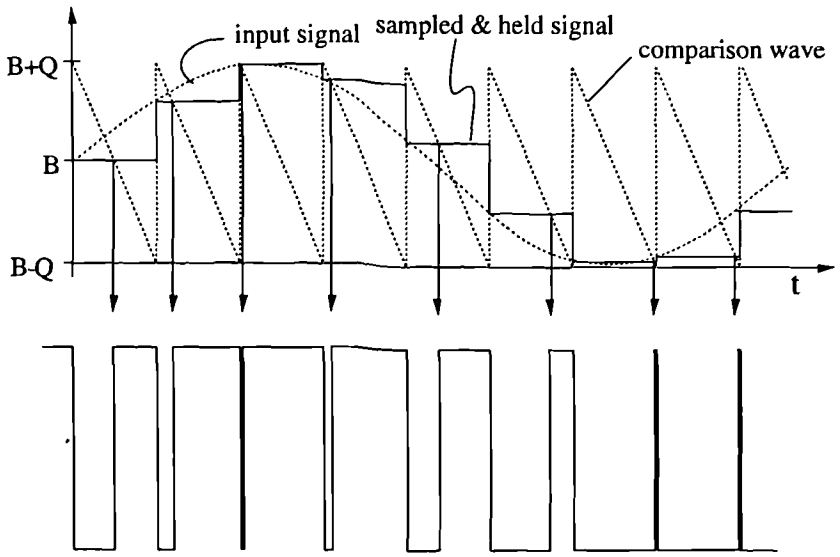


Figure 2.2.2.a : Sampling Waveforms at the Comparator Input Yielding Uniformly Sampled LEPWM

The digital implementation of TEPWM and LEPWM is relatively straightforward provided sufficiently low wordlength counters can be employed to avoid excessively high speed clock frequencies. Techniques to ensure this are discussed in chapter 4. Digital data sources usually provide regularly sampled data so uniformly sampled TEPWM can be produced by using the data values to define the 'mark' duration of the pulse. After loading this into a counter, the output can be 'set' and left high until the count is completed. The counter 'empty' can be decoded and used to 'reset' the output.

LEPWM can be created by subtracting the data value from the total sample period, and using the answer to define the low-duration. This can be achieved without signal processing by loading an up-counter with the sampled value and counting to 'full' to start the pulse. Since LEPWM has essentially the same performance as TEPWM but a more complicated circuit definition it is generally not used.

With appropriate manipulation of the modulation depth and offset the LEPWM and TEPWM outputs can be arranged to not overlap and hence by adding the outputs a two edged modulation with symmetry about the sampling instant can be created. This will be referred to as 'Symmetric pulse width modulation' or 'SYMPWM' for short. It should be noted that the sampling instant for each half of the pulse is the same (hence the symmetry) so like LEPWM and TEPWM, SYMPWM uses one sample per pulse.

In contrast, if naturally sampled TEPWM and LEPWM are manipulated in the same way to create one pulse, the sampling instants are not the same (one at each edge) hence SYMPWM cannot exist in a naturally sampled system.

2.2.3 : Basic Edge Position Algorithms using Two Samples per Pulse

If the TEPWM and LEPWM sawtooth waves are combined to make a triangle wave and used to perform natural sampling, a two sided modulation type is created with independent sampling instants for the leading and trailing edges. This modulation type will be referred to as 'double sided, pulse width modulation' or 'DSPWM' for short. The two sampling instants used to define each pulse cannot be taken from consecutive samples in a uniformly sampled data stream so DSPWM will be used to refer to two sided modulation only in a naturally sampled system.

If two consecutive samples are taken from a uniformly sampled signal another two sided modulation type can be created. One sample is used to define the duration of the leading portion, and the other to define the duration of the trailing portion. This will be referred to as 'two sample, consecutive, pulse width modulation' or '2SCPWM' for short. Clearly this modulation type depends on uniformly sampled input so 2SCPWM will be used to refer to two sided modulation only in a uniformly sampled system.

Both of these modulation schemes use two samples per pulse so the output PRR is half the input sample rate. When comparing the spectral performance of different PWM types, the carrier frequencies should be similar since it is this which combines with the signal to determine the sideband frequencies. This means that when comparing modulation types using two samples per pulse with those using one sample per pulse the input sample rates are not the same and the processing 'cost' to achieve the higher input sample rate must be taken into account.

In DPWM implementations using two samples per pulse, Fs (the digital sample rate) must equal 2.Fc (twice the carrier frequency). Representing two samples in one carrier period requires either the data resolution to be halved (one less bit) or the edge defining counters to run twice as fast. Counter frequency limits usually dictate that the edge resolution cannot be increased so more complicated signal processing is usually required (than for example, the single sided modulation types) to reduce the input wordlength by an extra bit, and this processing is operated faster for the increased sample rate.

2.2.4 : Special Edge Position Algorithms

For proper representation of SYMPWM, the leading portion should be of equal duration to the trailing portion. In a digital system the quantised data which takes on an odd value cannot be properly represented unless half-quanta time increments can be provided. This implies using twice the clock frequency in the edge defining counters. Since the clock frequency of the counters should be kept as low as possible this would suggest SYMPWM should not be used (although in the next section its spectral performance will be shown to be superior to the single sided modulation types).

To avoid using a higher clock rate, the pulses of odd valued duration can be shifted by half a clock cycle to re-align it to the clock edge. This effectively forces a slight asymmetry in some of the pulses. Where the pulse is shifted to occur later, its lags behind the ideal timing and this modulation type will be called 'lagging asymmetric pulse width modulation' or 'LAPWM' for short. The pulse could be shifted to occur earlier in time with a similar spectrum to LAPWM's but the circuitry for this is more complicated so it will not be considered further.

LAPWM exhibits very similar harmonic performance to true SYMPWM with half the clock rate (for the same carrier frequency) but the overall SNR is worse. To eliminate this problem, the asymmetry can be alternated or randomly placed to lead or lag the SYMPWM pulse position. The similarities of these techniques' performance will be shown by simulation and spectral analysis in section 2.3. The more complicated (from a circuit point of view) is the randomly shifted type and since this does not yield the best performance of the two, it will not be discussed further. The alternating odd-valued asymmetric pulse width modulation (AOAPWM for short) has harmonic performance similar to SYMPWM without the SNR reductions found in LAPWM or the circuit complexities of the randomly alternated asymmetric PWM. This will be demonstrated in detail by prototype and simulation and in general is used in place of SYMPWM for the benefits of its lower counter clock rate.

2.3 : Theoretical PWM Spectral Performance for Tonal Input.

2.3.1 : Overview

In this section the spectral performance of the various modulation types discussed in the last section will be presented. Since LEPWM and TEPWM can be shown to have the same spectral performance, that of TEPWM will be presented in place of both. Similarly, SYMPWM will be used not only to demonstrate its performance but also that of LAPWM and AOAPWM.

Derivation of arbitrary transfer functions are not easy since none of the modulation types are LTI processes. For this reason, the response to single tone inputs will be presented. The response to noise can be found for the white Gaussian case [ROW65], but since this is of little interest for most applications it will be omitted here. Continuous time, un-quantised signals will be used to simplify analysis since for highly oversampled discrete time, high resolution quantised systems, the errors involved are small. Such differences will be shown to be insignificant in chapters 5 and 6 where computer simulation (with spectral measurement) and prototype measurements demonstrate that the DPWM versions follow the 'analogue' theory presented here very closely.

The spectral response to tonal inputs can be found in most types of PWM by double Fourier analysis. This is presented in detail in appendix A.8, the conclusions of which are presented here. The analysis is possible for all the cases presented here by substituting Bessel functions for integrals that can only be found numerically. Without tonal input and edge position definitions that can be reduced to an appropriate form by substitution, the Bessel functions cannot be applied. This makes an expression for 'exotic' modulation types very complicated and performance can be predicted more easily by computer simulation. Appendix A.8 includes the preparation for numerical analysis of one such case which is used in chapters 5 and 6; the discussion of this case will be dealt with in those chapters.

In each spectrum presented, the modulating signal is :

$$\text{INPUT SIGNAL} = A_v . Cos(\omega_v . t) \qquad \text{Equation 2.3.1.a}$$

For this the modulation depth or index, M, can be defined as the ratio between the input signal amplitude, A_v , and the amplitude which causes modulation to the pulse period's limits, $A_{max.}$:

$$M = \frac{A_v}{A_{max.}} \qquad \text{Equation 2.3.1.b}$$

also the modulation offset, k is defined (to take into account DC offset between the input and output) :

$$k = \frac{\text{the unmodulated pulse duration}}{\text{the pulse repetition period}} \qquad \text{Equation 2.3.1.c}$$

In each spectrum presented, the signal harmonic terms will be indexed by 'n' and the carrier harmonic terms will be indexed by 'm'. 'J' will be used to represent the Bessel Function of the first kind and integer order as indicated by a subscript [ABR65].

2.3.2 : Naturally Sampled, Trailing Edged, Pulse Width Modulation

The output spectrum from naturally sampled, trailing edge modulation is the simplest of the PWM output spectra to derive, and is shown below :

$$F_{NTE}(t) = k + \frac{M}{2}.Cos(\omega_v t) + \sum_{m=1}^{\infty}\left\{\frac{1}{m\pi}.[Sin(m\omega_c t) - J_0(mM\pi).Sin(m\omega_c t - 2m\pi k)]\right\} - \sum_{m=1}^{\infty}\left\{\sum_{n=\pm 1}^{\infty}\left\{\frac{J_n(mM\pi)}{m\pi}.[Sin(m\omega_c t + n\omega_v t - 2m\pi k - \frac{n\pi}{2})]\right\}\right\}$$

Equation 2.3.2.a

The terms have been separated intentionally so that the first line is the sum of the DC content, the fundamental and harmonically related signal distortion components. The second line is the sum of harmonically related carrier distortion components and its fundamental. The last line is the sum and difference terms of the signal and its harmonics about the carrier and its harmonics; these make up the sideband distortion terms in the spectrum. All these terms have been plotted below to demonstrate a typical spectrum shape using a 2 kHz, full amplitude tone (M=1) and 44.1 kHz sample rate.

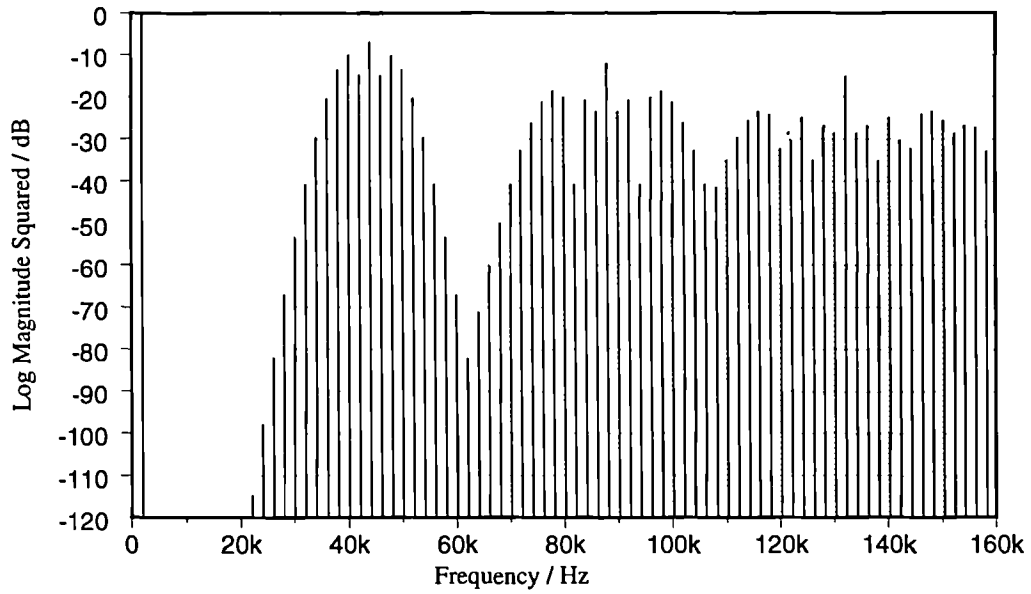


Figure 2.3.2.a : Theoretical Output Spectrum for Naturally Sampled TEPWM.

Several features should be noted from this spectrum : firstly, there is no harmonic distortion, only the input signal fundamental. Secondly, the signal sideband amplitudes fall away quickly, so for low frequency input signals the sideband distortion is well separated from the baseband. Thirdly, the carrier harmonics and their associated sidebands reduce in amplitude with increasing frequency so the power in the carrier fundamental is the most significant part wasted in high frequency spectral components. Lastly, the sideband distortion components are spaced by the input frequency (and integer multiples of it) from the carrier. This implies that only particular frequency bands in the input can produce distortion which result in sidebands recurring in the signal band.

2.3.3 : Uniformly Sampled, Trailing Edged, Pulse Width Modulation

The output spectrum from uniformly sampled, trailing edge modulation can be derived in a similar way to the natural case by a change of variable and new integration limits to find the Fourier coefficients (see appendix A.8). The new spectrum is shown below :

$$F_{UTE}(t) = k - \sum_{n=1}^{\infty}\left\{\frac{J_n\left(\frac{n\pi M\omega_v}{\omega_c}\right)}{\frac{n\pi\omega_v}{\omega_c}} Sin\left(n\omega_v t - \frac{2n\pi k\omega_v}{\omega_c} - \frac{n\pi}{2}\right)\right\}$$

$$+ \sum_{m=1}^{\infty}\left\{\frac{1}{m\pi}.[Sin(m\omega_c t) - J_0(mM\pi).Sin(m\omega_c t - 2m\pi k)]\right\}$$

$$- \sum_{m=1}^{\infty}\left\{\sum_{n=\pm 1}^{\pm\infty}\left\{\frac{J_n[(m\omega_c + n\omega_v)\frac{\pi M}{\omega_c}]}{(m\omega_c + n\omega_v)\frac{\pi}{\omega_c}} Sin\left[(m\omega_c + n\omega_v)\left(t - \frac{2\pi k}{\omega_c}\right) - \frac{n\pi}{2}\right]\right\}\right\}$$

Equation 2.3.3.a

From this the harmonic distortion terms in the spectrum can clearly be seen in the revised first line. The remaining terms are similar to the natural case (although slightly lower in amplitude) and with different phase distortion terms. It should also be noted that the signal response is no longer constant for all frequencies with both amplitude and phase error. A plot of this theoretical spectrum is shown below with the same signal and carrier parameters as before to show the additional harmonic distortion terms.

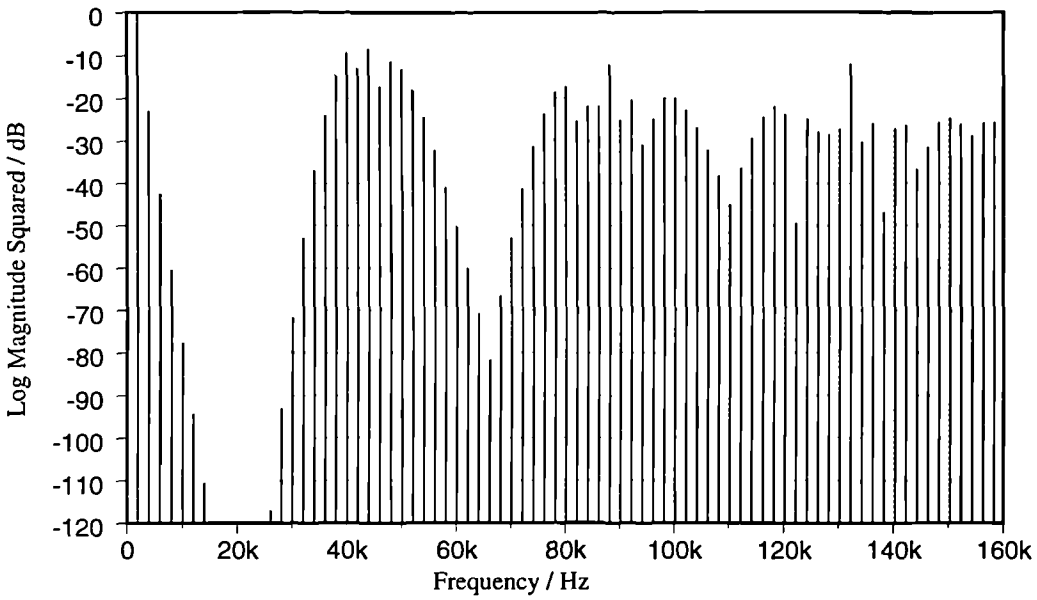


Figure 2.3.3.a : Theoretical Output Spectrum for Uniformly Sampled TEPWM.

The distortion introduced by uniformly sampling the signal makes this type of modulation less suitable for highly linear applications such as audio. Reducing the modulation depth or increasing the signal-carrier frequency ratio helps to lower the distortion, but not by a great deal as will be shown in sections 2.4.2 and 2.4.3 . The reduced sideband distortion amplitude compared to the naturally

sampled version is significant (although small) because this would indicate a lower sensitivity to high frequency input noise being modulated into the signal band. This point will be returned to later.

2.3.4 : Symmetric (Uniformly Sampled) Pulse Width Modulation

Taking the sum of a uniformly sampled, TEPWM and uniformly sampled, LEPWM with each modulated to half the depth with half the offset yields the spectrum of SYMPWM. In this case the spectra can be added together because by adjusting the depth and offset, the output pulses cannot overlap resulting in a pulse which has both its sides controlled by the same sample. The spectrum of LEPWM can be derived from TEPWM's by simply inserting '-t' in place of 't' in the original spectrum (see appendix A.8). After collating terms the SYMPWM spectrum is as shown below :

$$F_{SYM}(t) = k + \sum_{n=1}^{\infty} \left\{ \frac{2J_n(n\frac{\omega_v}{\omega_c}\pi\frac{M}{2})}{n\pi\frac{\omega_v}{\omega_c}} Cos(n\omega_v t).Sin(n\pi k\frac{\omega_v}{\omega_c}+\frac{n\pi}{2}) \right\}$$
$$+ \sum_{m=1}^{\infty} \left\{ \frac{2J_0(m\pi\frac{M}{2})}{m\pi} Cos(m\omega_c t).Sin(m\pi k) \right\}$$
$$+ \sum_{m=1}^{\infty} \left\{ \sum_{n=\pm1}^{\pm\infty} \left\{ \frac{2J_n[\pi\frac{M}{2}\frac{(m\omega_c+n\omega_v)}{\omega_c}]}{\pi.\frac{(m\omega_c+n\omega_v)}{\omega_c}} Cos[(m\omega_c+n\omega_v)t] Sin[\frac{m\omega_c+n\omega_v}{\omega_c}\pi k+n\frac{\pi}{2}] \right\} \right\}$$

Equation 2.3.4.a

Like other uniformly sampled modulation types, the output contains harmonic distortion as well as a DC offset, the carrier, its harmonics and sideband terms. Compared to uniformly sampled TEPWM's spectrum, the signal-harmonic distortion is dramatically reduced (see below); furthermore, when the product of 'm' and 'k' is an integer, the carrier harmonic will be absent as a result of phase cancellation (eg. for k=0.5, even index carrier terms are cancelled).

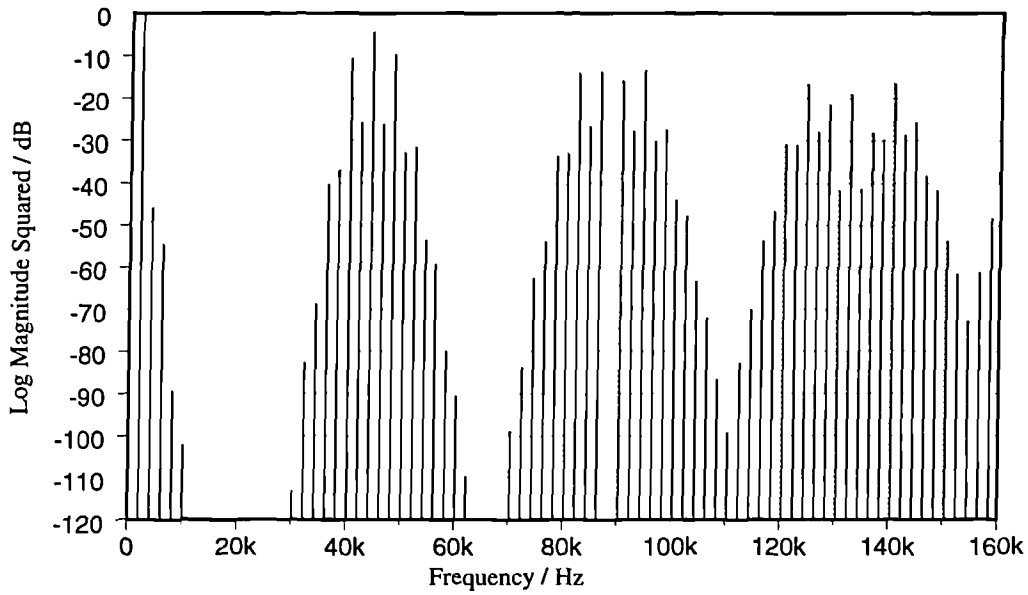


Figure 2.3.4.a : Theoretical Output Spectrum for SYMPWM.

As mentioned before, the harmonic performance of LAPWM and AOAPWM is very similar to SYMPWM and this is demonstrated below (by simulation) with the same parameters as used above.

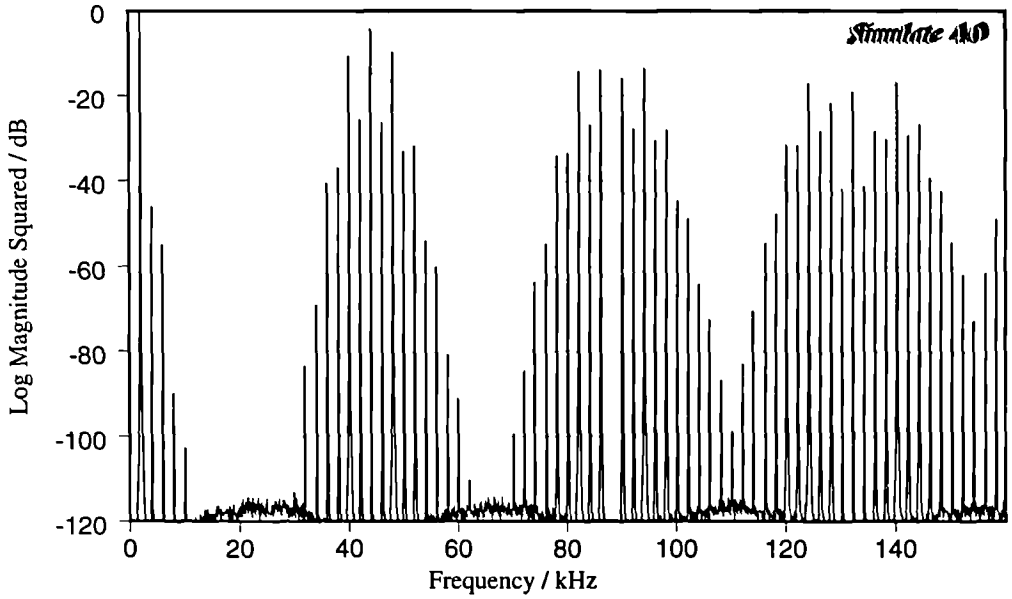


Figure 2.3.4.b : Simulated Output Spectrum for LAPWM.

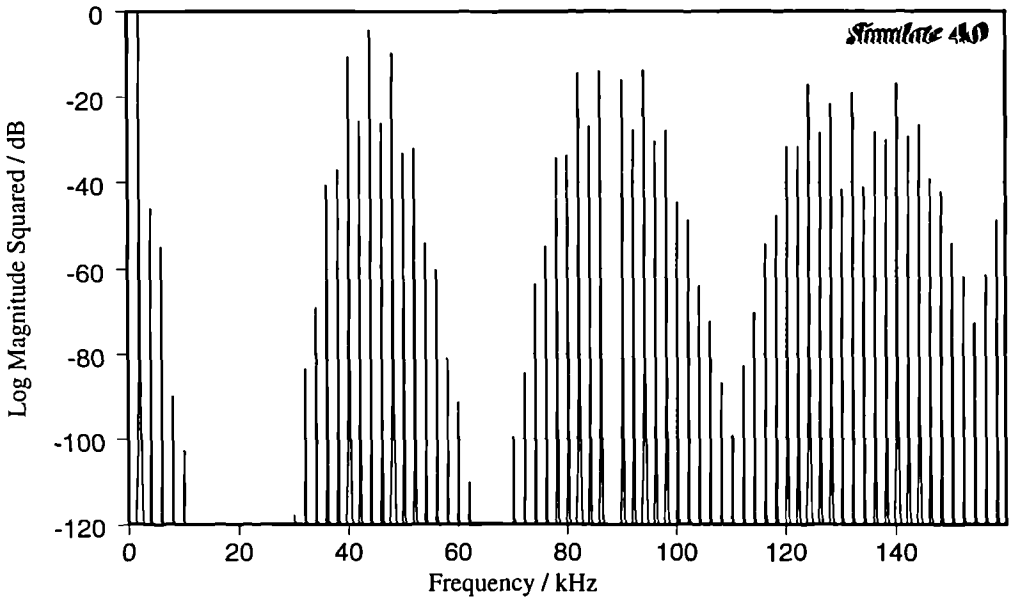


Figure 2.3.4.c : Simulated Output Spectrum for AOAPWM.

Both LAPWM and AOAPWM use half the internal clock rate of that used for producing SYMPWM. Nb.: additional noise is introduced by LAPWM, so AOAPWM will be preferred.

2.3.5 : Double Sided (Naturally Sampled) Pulse Width Modulation

DSPWM can be derived from naturally sampled TEPWM using a similar technique to the derivation of the SYMPWM spectrum from the uniformly sampled TEPWM spectrum. Again, the

leading edged version of the spectrum has to be found, the modulation depth and offset have to be halved, the two spectra added and the terms collated. This yields DSPWM's spectrum as shown below :

$$F_{DSN}(t) = k + \frac{M}{2} Cos(\omega_v t) + \sum_{m=1}^{\infty} \left\{ \frac{J_0(m M \frac{\pi}{2})}{m \frac{\pi}{2}} Cos(m \omega_c t) Sin(m \pi k) \right\} + \sum_{m=1}^{\infty} \left\{ \sum_{n=\pm 1}^{\infty} \left\{ \frac{J_n(m M \frac{\pi}{2})}{m \frac{\pi}{2}} Cos(m \omega_c t + n \omega_v t) Sin(m \pi k + n \frac{\pi}{2}) \right\} \right\}$$

Equation 2.3.5.a

Like other naturally sampled modulation types, there is no harmonic distortion term present in the output, although the familiar group remain (DC offset, carrier, its harmonics and sideband terms). When compared to naturally sampled TEPWM, the output has fewer sideband components, indicating a smaller tendency to produce sidebands in the baseband from high frequencies in the input. Fewer sidebands arise from phase cancellation as found in the SYMPWM case (2mk+n, even). Similarly, some carrier harmonics are absent (m.k, integer). A theoretical output spectrum is plotted below (k=0.5):

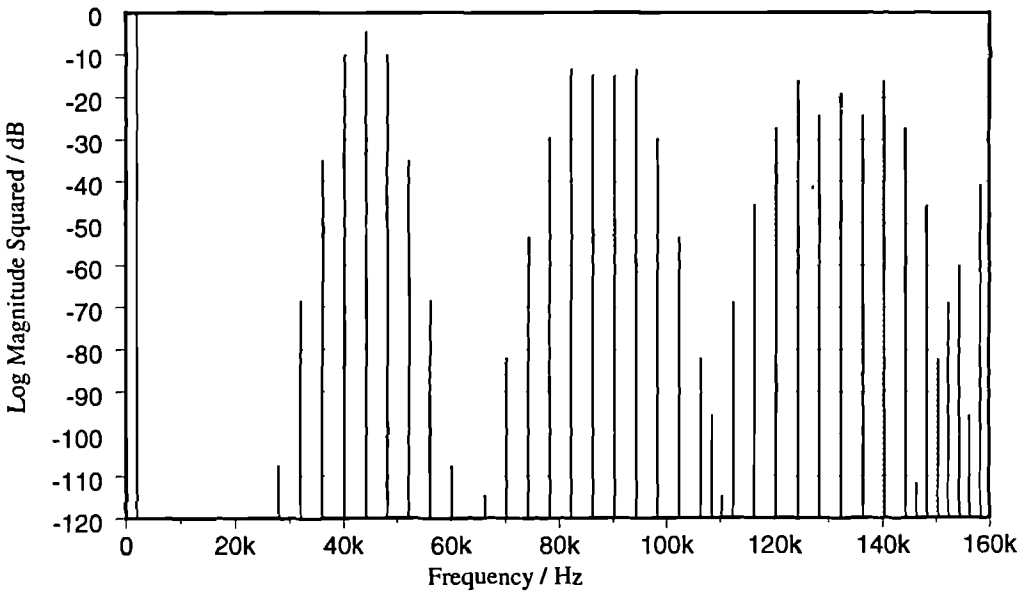


Figure 2.3.5.a : Theoretical Output Spectrum for DSPWM.

2.3.6 : Two Sample Consecutive (Uniformly Sampled), Pulse Width Modulation

As described in section 2.2.3, 2SCPWM can be modelled as the sum of uniformly sampled TEPWM and LEPWM. Its spectral performance can be found taking the spectrum of each part with half the modulation depth and half the modulation offset, and inserting a half carrier period delay to the LEPWM signal compared to the TEPWM signal. The spectrum so found is particularly interesting for phase cancellation of *all* even ordered signal-harmonic terms. In the case when the unmodulated pulse is of 50% duty cycle (k=0.5), and the input signal has no DC offset, the term "(2k-1)" becomes zero, so that the amplitudes of harmonics of the signal and some of the sidebands become scaled by "Sin(nπ/2). Thus even ordered signal distortion terms in the baseband and some sidebands are absent :

$$F_{2SC}(t) = k + \sum_{n=1}^{\infty}\left\{\frac{2J_n(n\pi\frac{M}{2}\frac{\omega_v}{\omega_c})}{n\pi\frac{\omega_v}{\omega_c}} Cos(n\omega_v t - n\frac{\pi}{2}\frac{\omega_v}{\omega_c}) Sin(n\frac{\pi}{2}[(2k-1)\frac{\omega_v}{\omega_c}+1])\right\}$$

$$+\sum_{m=1}^{\infty}\left\{\frac{2}{m\pi} Cos(m\omega_c t - m\frac{\pi}{2}).[Sin(m\frac{\pi}{2}) + J_0(m\frac{M}{2}\pi) Sin([2k-1]m\frac{\pi}{2})]\right\}$$

$$+\sum_{m=1}^{\infty}\left\{\sum_{n=\pm1}^{\infty}\left\{\frac{2J_n[\pi\frac{M}{2}\frac{(m\omega_c+n\omega_v)}{\omega_c}]}{\pi\frac{(m\omega_c+n\omega_v)}{\omega_c}} Cos[(m\omega_c+n\omega_v)(t-\omega_c\frac{\pi}{2})] Sin[\frac{\pi}{2}(2k-1)\frac{m\omega_c+n\omega_v}{\omega_c} - n\frac{\pi}{2}]\right\}\right\}$$

Equation 2.3.6.a

When compared to SYMPWM, the total harmonic distortion figures are very similar (although alternately phase modulated by ±1 and concentrated in odd order terms). Since the even harmonics are absent, the distortion is effectively at a higher frequency so the audible severity of this distortion is smaller. As will be shown in the next section, the trend for distortion arising from PWM is for it to be more severe for high frequency input signals. This feature gives 2SCPWM a significant linearity advantage since the input frequencies that PWM most severely distorts (high frequencies), will have their most severe harmonic term outside the audio band, enabling a significant average THD reduction when evaluated over the audio band. If psychoacoustic weighting is included in this comparison, 2SCPWM appears even more favourable since high frequency distortion is less significant after weighting (eg.: E-weighted [VAN89]). The theoretical output spectrum is plotted below :

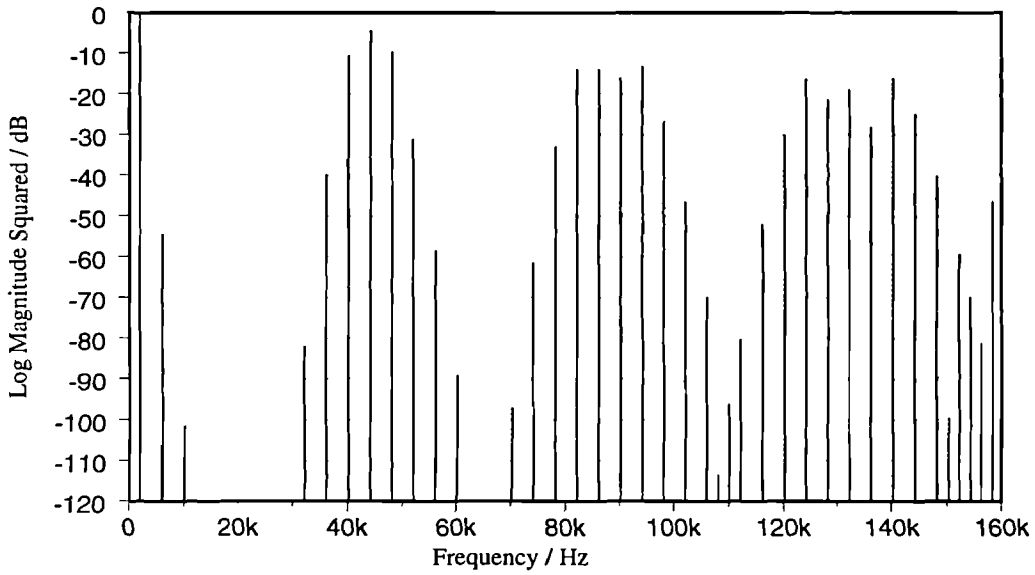


Figure 2.3.6.a : Theoretical Output Spectrum for 2SCPWM.

As with other uniformly sampled PWM types, harmonic distortion is present and in that sense the modulation type is less useful than the naturally sampled types. The computational complexity of producing such an output is not severe, so this type of modulation is reasonably suited to power amplifier applications where total linearity may not be possible anyway due to other circuit non-idealities. The sideband amplitudes are not as small as in SYMPWM but they are smaller than in uniformly sampled TEPWM, so where high frequency input noise cannot be avoided and reasonable linearity is required this modulation type should be favoured over other uniformly sampled types.

The cancellation of even order harmonic distortion and some sideband terms is particularly sensitive to the accuracy of the modulation offset, k. Theoretically, at these frequencies and for k=0.5, there is no distortion, but in practical systems where errors in this value can occur, cancellation may not be perfect. Such errors arise from differences in the rising and falling edge shape, differences in the switching delay times, or DC signal content, so the theoretical performance of modulation types depending on such cancellation is difficult to achieve in practise.

To assess the significance of varying 'k', a summary graph of the second harmonic level plotted against 'k' is shown below for some values near k=0.5 . For this series of simulations, a worst case tone of 10 kHz @ -1.5 dBFS was used in a ten times oversampled system. The level of second harmonic distortion using a uniformly sampled, single sided, modulation type would have been -37.0 dBFS under these conditions, or -71.5 dBFS for a uniformly sampled, two sided modulation type. As can be seen from figure 2.3.6.b (below) 2SCPWM is still better than a single sided modulation type but approaches the performance of a two sided modulation type depending on the accuracy to which k=0.5 can be maintained.

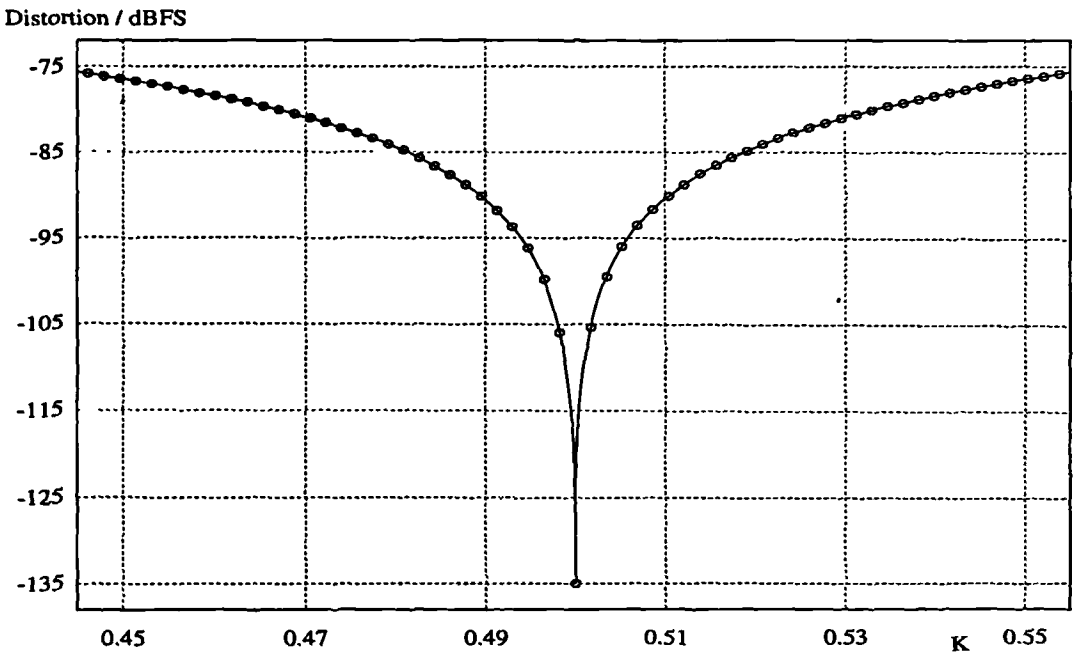


Figure 2.3.6.b : Variation in 2 nd. Harmonic Level vs. Modulation Offset, 'k', for 2SCPWM.

2.4 : PWM Distortion Trends and Relative Performance.

2.4.1 : Overview

To use pulse width modulation as part of a high quality D/A converter the modulator should be capable of low noise and highly linear performance. Using a digital implementation of a PWM can avoid difficulties which the analogue counterparts are prone to such as electrical noise but the modulation process itself presents difficulties. None of the modulation variants presented in the last sections are linear, time-invariant processes. In every case, the modulation introduces sidebands which may be modulated into the signal band; in all the uniformly sampled cases harmonic distortion is also introduced.

With all the modulation types the distortion component amplitudes vary with modulation index, 'M', modulation offset, 'k', the harmonic indices, 'm' and 'n', and the signal to carrier frequency ratio, 'ω_v/ω_c'. Varying 'M' and 'ω_v/ω_c' strongly affects the level of harmonic and sideband terms so the relationships between these parameters and the resulting distortion severity will be examined in sections 2.4.2 and 2.4.3, to decide how to set up an acceptable conversion process.

Different modulation variants exhibit different levels of harmonic distortion under similar test conditions. These will be compared in section 2.4.4 to illustrate the benefits that can be gained by choosing a less distorting variant which may then allow relaxation of one of the other parameters. Similarly, sideband performance forms a limit on the performance that can be achieved from any given PWM type for chosen 'M' and 'ω_v/ω_c'. The different types exhibit different sensitivity to high frequency signals which might occur at frequencies which are modulated into the signal band. A comparison of the theoretical sensitivity of such tones is presented in section 2.4.5.

Although not obvious from the tonal responses, the uniform modulation types produce intermodulation between signals in the baseband. This can be demonstrated by simulation, and a comparison of the severity of such intermodulation, using twin-tone input is presented in section 2.4.6.

2.4.2 : Distortion Reduction with Modulation Depth

All the spectra presented in section 2.3 showed distortion amplitudes for both harmonic terms and sideband generated terms of the general form :

$$\text{Distortion Amplitude } \alpha \frac{J_x(M.y)}{y} \qquad \text{where } y = f\{m, n, \omega_c, \omega_v\} \qquad \text{Equation 2.4.2.a}$$

where 'x' is the carrier or signal distortion component index ('m' or 'n'). The Bessel functions J_1 to J_4 are plotted below, where it can be seen that for decreasing argument the Bessel function value decreases provided the argument is small. This implies that the distortion decreases as the modulation depth decreases, whether this is reduced by lowering the modulation index or by using a smaller input signal amplitude. Also from this graph, the gradients of the distortion amplitude terms ($x > 1$) can be seen to be increasing so reductions in the modulation depth lead to diminishing reductions in the distortion :

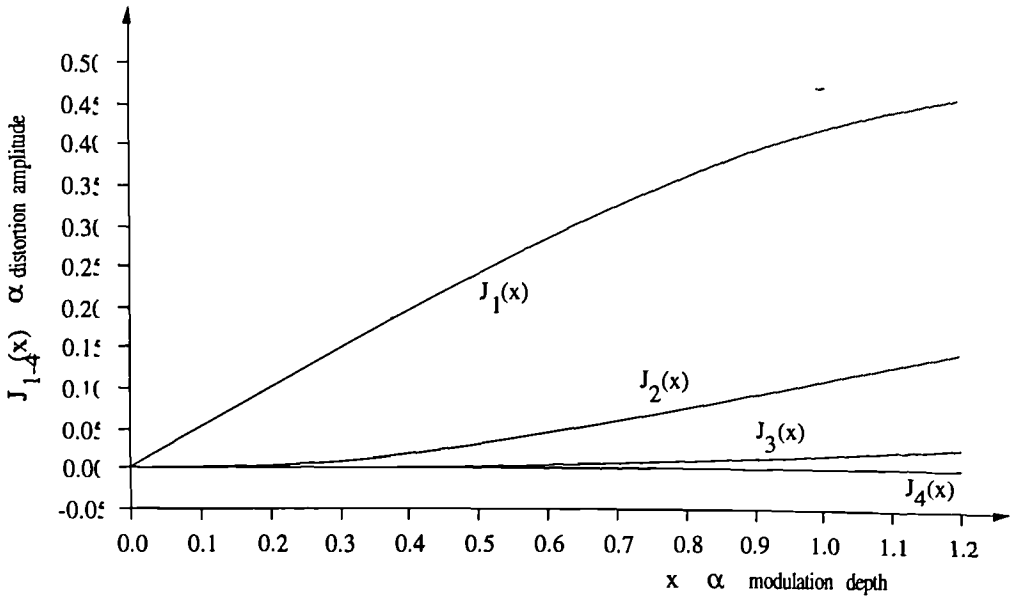


Figure 2.4.2.a : Bessel Functions J_1 to J_4. †

Measurements taken from simulations confirm the trends shown above; shown below is the dominant harmonic distortion (2 nd.) plotted against modulation depth for uniformly sampled trailing edged modulation, a 6 kHz input tone and a pulse repetition rate of 352.8 kHz.

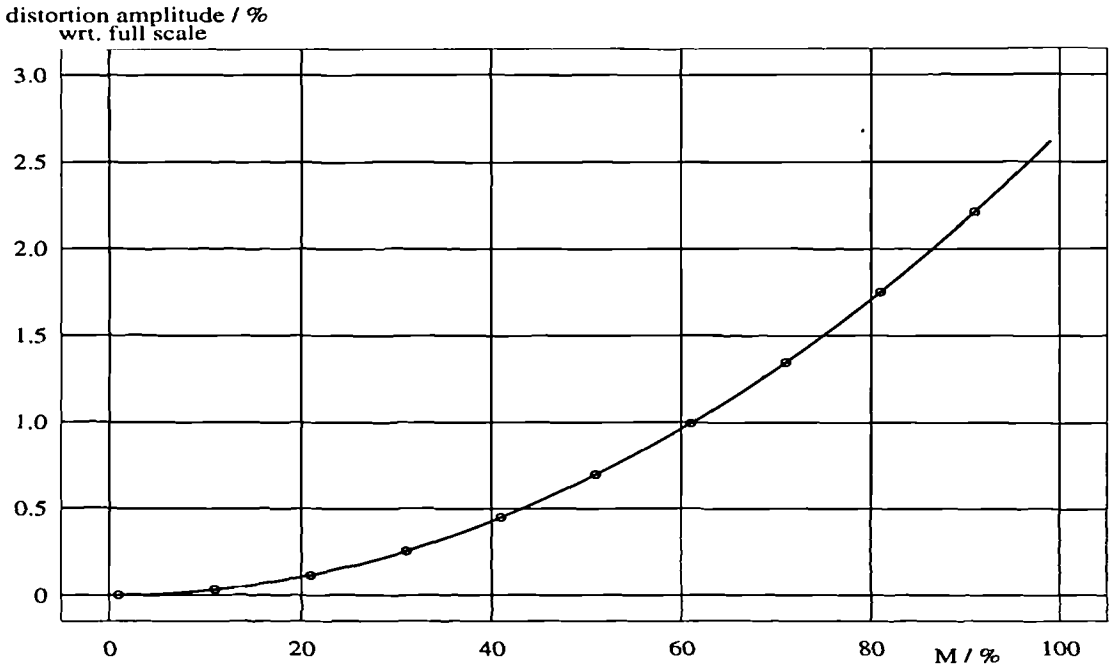


Figure 2.4.2.b ; Variation in Dominant Distortion Level vs. Modulation Depth for TEPWM

2.4.3 : Distortion Reduction with Signal-Carrier Frequency Ratio

From equation 2.4.2.a, the distortion levels arising from PWMs can be seen to be a function of the denominator 'y' although this relationship is more complicated than for the modulation depth. By plotting, the functions '$J_{1-4}(x)/x$' the variation of distortion amplitude with the signal-carrier

† This "x" represents the Bessel function's *argument*, and is proportional to the modulation depth because other factors have been held constant for this set of calculations; this is not the same as "x" as used in equation 2.4.2.a, which represents the *order* of the Bessel function.

frequency ratio ('ω_v/ω_c') can be seen; from this the reduction in distortion can be predicted for the uniformly sampled modulation types when using a high pulse repetition rate (oversampling).

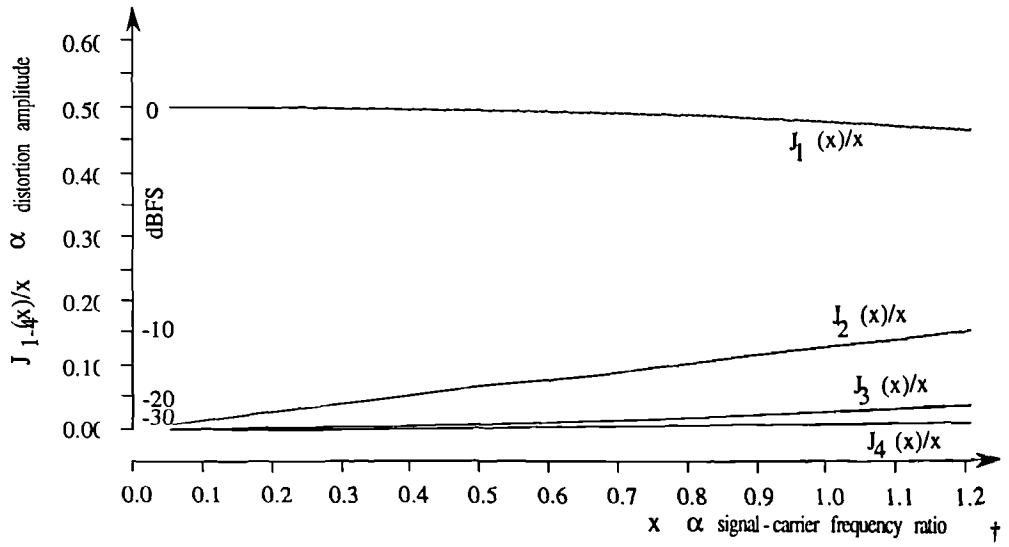


Figure 2.4.3.a : Variation in Uniformly Sampled PWM Distortion with Signal-Carrier Frequency Ratio

It is worth noting from the above graph that as the signal to carrier frequency ratio diminishes, both the harmonic distortion reduces ($J_{2\text{-}4}(x)/x$) and the signal amplitude is modified less ($J_1(x)/x$). Signal phase distortion and sideband distortion amplitude also depend on 'ω_v/ω_c' in a similar way so oversampling has been recommended for making PWM linear [SAN83].

The above graph is also useful for visualising the proportions of distortion contribution that arise from each harmonic term (ie. 2 nd., 3 rd., 4 th. ...). From equation 2.3.3.a, for the worst case signal amplitude (ie. M=1), x can be seen to be :

$$x = \frac{n.\pi.\omega_v}{\omega_c}$$ Equation 2.4.3.a

Noting that the amplitude of harmonic distortion terms becomes larger for higher frequency input tones, the worst case distortion frequency can be deduced to be at the high frequency end of the band of interest; for the audio band this will be assumed to be 20 kHz. The frequency at which the n^{th} harmonic of a signal at ω_v is simply $n.\omega_v$, so for this to lie at 20 kHz, and ω_c converted to kHz :

$$x = \frac{20.\pi}{F_c}$$ Equation 2.4.3.b

(ie. x=1.2, implies F_c=52 kHz, x=0.178 implies F_c=352.8 kHz, x=0.089 implies F_c=705.6 kHz, etc.).

2.4.4 : Harmonic Distortion Comparison across Modulation Types

When comparing the harmonic distortion arising from the various modulation types there are three main points to note which can be seen in the theoretical spectral performance.

Firstly, the naturally sampled modulation types do not produce harmonic distortion. The dominant distortion terms in the naturally sampled modulation types are the sideband terms which can

† This "x" represents the Bessel function's *argument*, and is proportional to the signal-carrier frequency ratio because "M" has been set to unity and "m" & "n" have been held constant for this set of calculations; this is not the same as "x" as used in equation 2.4.2.a, which represents the *order* of the Bessel function.

be avoided by using a high pulse repetition rate so that when the sidebands are large enough to be significant they are at too high a frequency to be audible.

Secondly, when comparing the uniformly sampled modulation types (essentially TEPWM and SYMPWM or 2SCPWM), the single sided modulation's distortion terms are not scaled down by a "Sine" term. Furthermore, the Bessel function argument 'y' (as used in equation 2.4.2.a) is doubled in the single sided modulation case, having an effect similar to doubling the signal-carrier frequency ratio, 'ω_v/ω_c'. Both these differences lead to far larger distortion amplitudes in the single sided case.

Thirdly, 2SCPWM with appropriate modulation offset, 'k', has no even order distortion terms. For k=0.5, the "Sine" multiplier in the harmonic distortion summation of equation 2.3.6.a, simplifies so that for harmonics or sideband terms with 'n' even, the harmonic distortion becomes zero:

$$Sin\left(n\frac{\pi}{2}\left[(2k-1)\frac{\omega_v}{\omega_c}+1\right]\right) \longrightarrow Sin\left(n\frac{\pi}{2}\right)$$

Equation 2.4.3.a

Shown below is a graph summarising the THD level (arising from distortion up to 20 kHz) plotted against input signal frequency using from a 50% modulated tone and a pulse repetition rate of 352.8 kHz with each of the uniformly sampled modulation types.

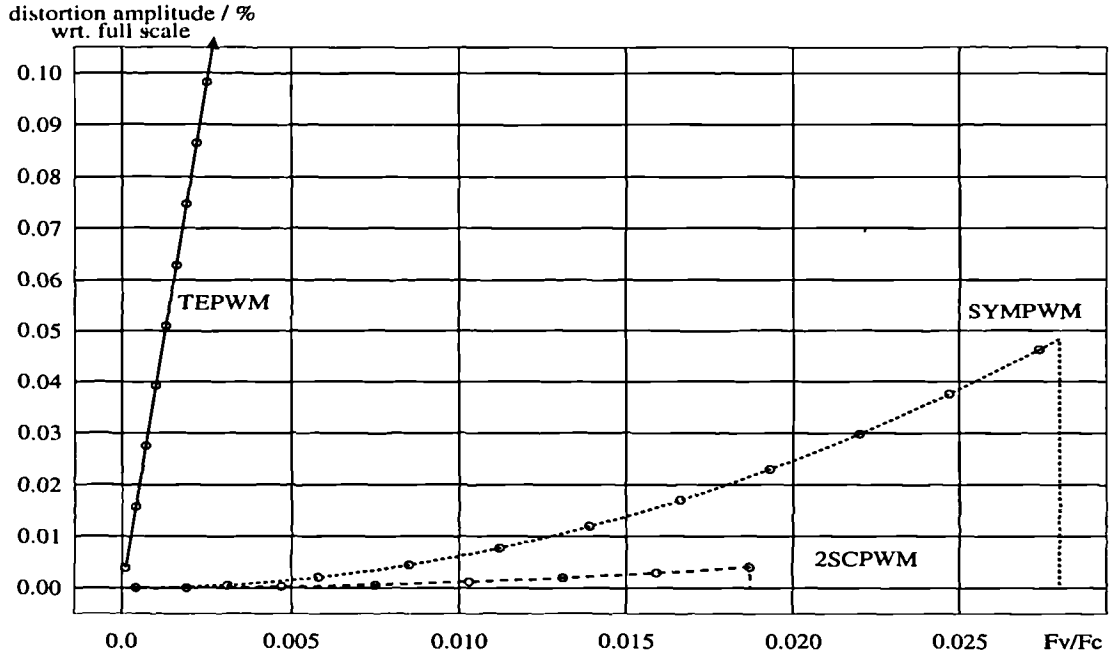


Figure 2.4.4.a : THD vs. Input Signal Frequency for Various Modulation Types

In the above graph, the two sided modulation types can be seen to be significantly better than the single sided one, and a large reduction in the THD in the 2SCPWM case can be seen when the third harmonic term becomes above 20 kHz and thus 'out of band'.

2.4.5 : Sideband Performance Comparison across Sampling Types

Sidebands arise from all the modulation types discussed and where these are of significant size in the signal band, the THD will be increased. Both the harmonic and sideband distortion terms fall rapidly with harmonic index 'n' so if sufficient oversampling is used, the sideband terms can be made

arbitrarily small in the signal band. Since the sideband terms fall rapidly with index, only those arising from the carrier fundamental need be designed for; higher carrier harmonics will produce far smaller in-band distortion because of the larger harmonic index required to place the sideband in audible range.

In contrast to the harmonic distortion 'order of merit', the naturally sampled modulation types produce worse sideband distortion than the uniformly sampled ones, although as before, the two sided modulation types produce less distortion than the single sided modulation types. As a result of 2SCPWM producing only odd order sideband terms (as discussed in section 2.4.4) the distortion from this modulation type is particularly small.

To compare the relative merits of all the modulation types, a graph of 'safe operating area' can be plotted for tonal input, showing the frequencies and amplitudes which will not produce in-band sidebands. Shown below is an example plot requiring that the in-band components be smaller than -120 dBFS (approximately 16 bit quantisation noise floor level). Note that wide areas of insensitivity exist because out of band distortion can be ignored.

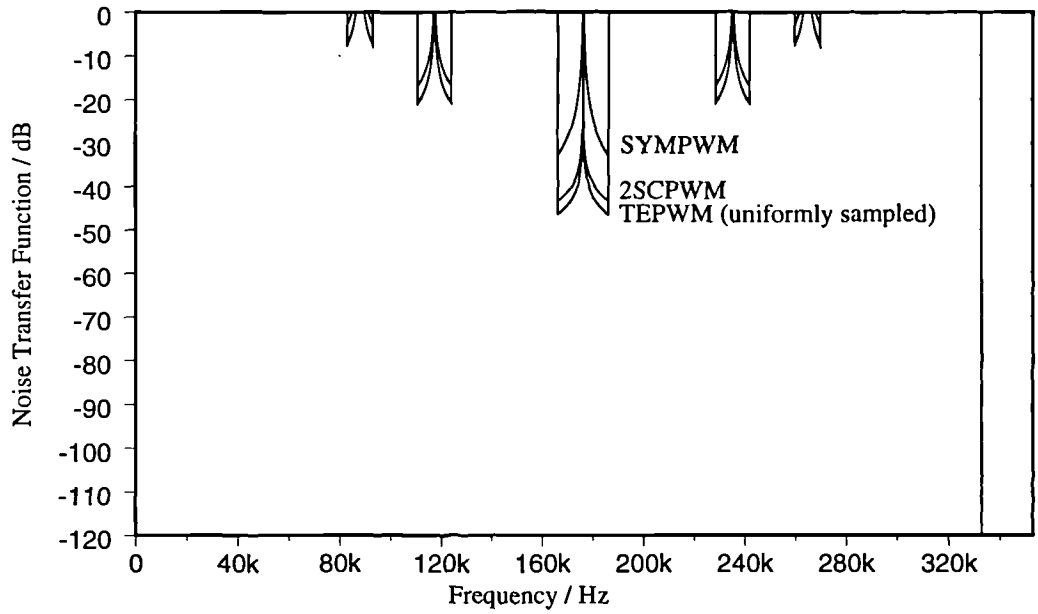


Figure 2.4.5.a : 'Safe Operating Areas' for Various Modulation Types in a PRR=352.8 kHz System

The above graph is particularly useful for gauging the relative merits of the various modulation types but this is only the performance with single tones as input. If the high frequency input content is noise-like the levels which cause sidebands in the audio band should be revised. No formal analysis is known for this but extensive computer simulation suggests that an additional 10 to 20 dB of tolerance can be expected. This uncertainty increases the importance of specific computer simulation of a proposed system to eliminate reduced performance by modulated noise appearing in the signal band.

2.4.6 : Intermodulation Distortion Comparison across Modulation Types

Although the theoretical spectral performance has been carried out relying on the single tonal responses, further intermodulation between input signal components produces distortion at sum and

difference frequencies. Like harmonic distortion, naturally sampled modulation types do not produce this distortion so shown below are spectra for only uniformly sampled modulation types. Test signals of 50% modulation depth, 3 & 4 kHz input tones are used and intermodulation can clearly be seen at 1 and 7 kHz (nb.: distortion at 6, 8, 9, 12 kHz etc. is not intermodulation but harmonic distortion).

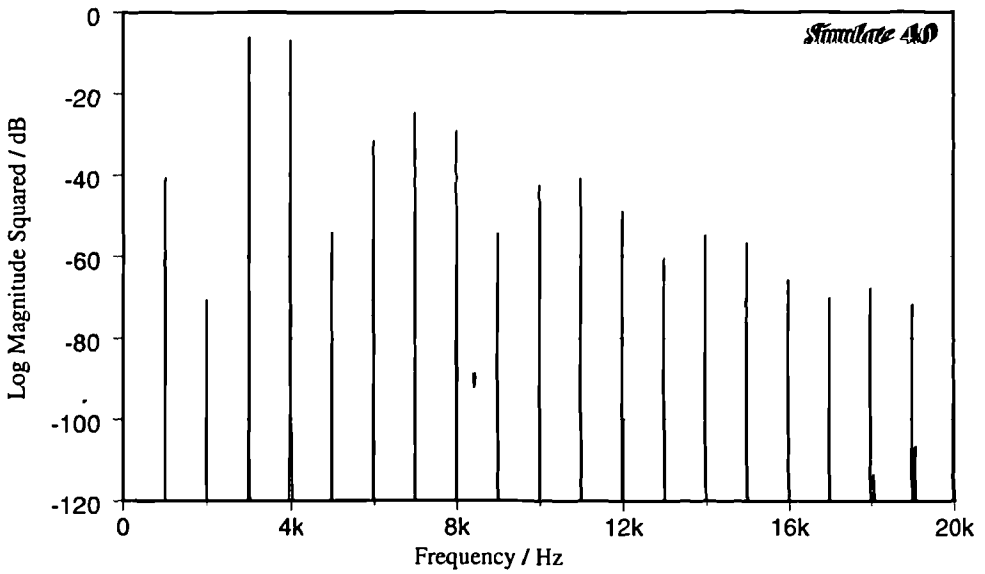


Figure 2.4.6.a : Intermodulation at 1 & 7 kHz arising from Uniformly Sampled TEPWM

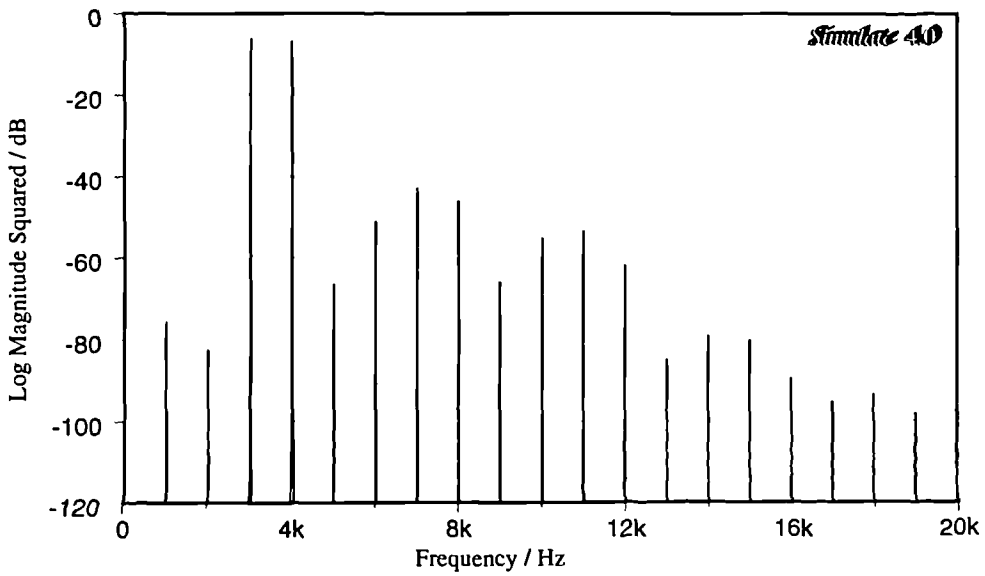


Figure 2.4.6.b : Intermodulation at 1 & 7 kHz arising from SYMPWM

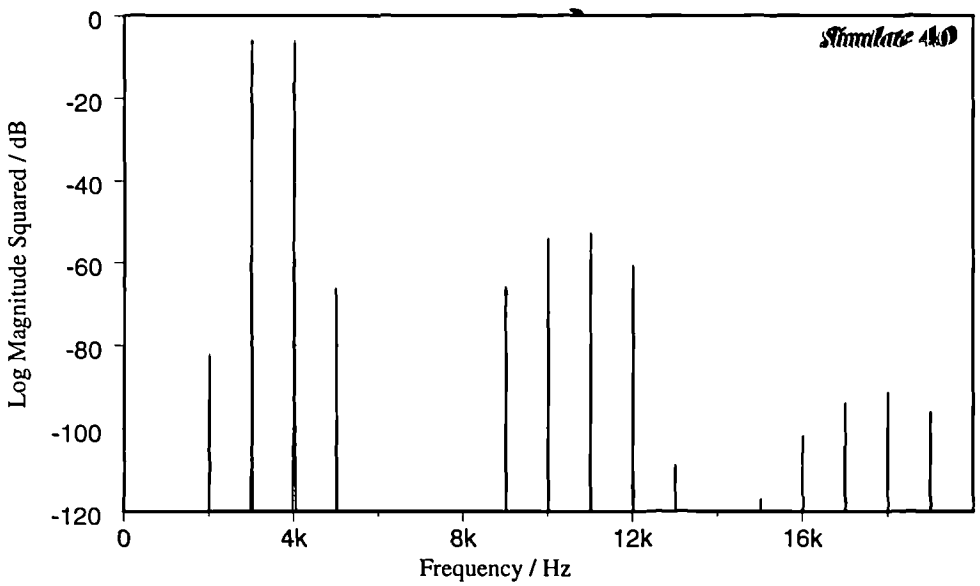


Figure 2.4.6.c : Intermodulation at 1 & 7 kHz arising from 2SCPWM

It should be noted that the 2SCPWM distorts in this way less than the SYMPWM, which, in turn, is better than the TEPWM. This is not only of concern for tones in the input, but also for noise near the end of the audio band. Although prior to modulation, frequencies just outside the 0-20 kHz band would not contribute to an SNR measurement, after modulation these frequencies can reappear in the audio band. This fact will later be the reason for more aggressive signal processing in the noise shaping circuitry (see chapter 4) since more than just the audio band has to kept clear of noise. Although not illustrated here, other uniformly sampled two sided modulation types (LAPWM and AOAPWM) also exhibit similar performance to SYMPWM and provision should be made to avoid this problem in these modulation types as well.

2.5 : Summary.

Seven specific algorithms for modulating the edges of a pulse have been examined in detail and all of these have been shown to produce some form of distortion. The algorithms fall into two categories according to the method of sampling used : natural or uniform. Uniformly sampled modulation types are well suited for use with uniformly sampled data (as used in PCM systems) and these can be easily constructed in analogue or digital circuits.

Naturally sampled trailing edged modulation (TEPWM) and double sided modulation (DSPWM) both exhibit no harmonic distortion and no intermodulation between separate parts of the signal. The carrier frequency, its harmonics and frequency modulated sidebands are present in the output and the amplitude of these terms when at frequencies within the signal band is considered to be the limiting factor for the linearity that these modulation types can offer.

Uniformly sampled TEPWM, symmetric modulation (SYMPWM) and modulation produced by defining leading and trailing edge position from consecutive regular samples (2SCPWM) all produce distortion. This arises harmonically related to the signal and carrier, as intermodulation between different parts of the signal, as signal sidebands about the carrier and its harmonics and as amplitude modification of the signal itself. The harmonic distortion arising from uniformly sampled single sided PWM is substantially larger than that produced by the other uniformly sampled modulation types.

Distortion in uniformly sampled modulation types and sidebands in all the modulation types can be reduced by using a smaller modulation depth or a higher ratio between the signal and carrier frequencies. Thus, high frequency input signals at high amplitude are the most severely distorted part of any input signal and oversampling can be used to restrict this effect. The sidebands of uniformly sampled modulation types are smaller than those produced by naturally sampled modulation types which may permit higher input noise levels at high frequencies without damaging the output signal band with noise sidebands.

2SCPWM can be forced to have no even order harmonics by using a 50 % duty cycle in the absence of a modulating signal. This allows significant amounts of distortion to be ignored because it will be at sufficiently high frequency to be less audible, giving this modulation type an apparent linearity advantage over SYMPWM.

Two further uniformly sampled two sided modulation types can be used to reduce the edge resolution required in a pulse width modulator by a factor of two; these are unique to digital PWM implementations. By delaying odd valued SYMPWM pulses the edges can be realigned with a lower frequency clock in the DPWM allowing more feasible counting rates. This does introduce slight pulse asymmetry hence the modulation name : lagging asymmetric modulation (LAPWM).

LAPWM produces additional noise (compared to SYMPWM) which can be reduced by alternating the side on which the asymmetry is placed (ie. alternately advancing or delaying odd valued pulses); this modulation type is called alternating odd asymmetry modulation (AOAPWM). Both AOAPWM and LAPWM have been found by simulation to perform harmonically in the same way as SYMPWM so distortion comparison with SYMPWM can be used in place of individual analysis.

Chapter 3 : Multi-Rate Filtering

3.1 : Multi-Rate Filtering Fundamentals

3.1.1 : Overview

Multi-rate filtering is a subset of the general implementation of digital filters which can carry out frequency-dependent, linear processing in a complex sense (ie. control of magnitude or phase as functions of frequency). As in analogue filtering, the performance of a filter network can be predicted by nodal analysis, leading to a transfer function with complex variables. For causality in a digital system no node can be expected to be evaluated without delay and hence the minimum delay in any loop is one cycle of a synchronising clock.

Digital filtering splits largely into two families, one with finite impulse response (FIR) and one with infinite impulse response - IIR (although its response may decay to insignificance in a shorter time). Digital filters can be used without linear, time invariant (LTI) filter implementations or regular sampling of signal data, but both will be assumed for the ideas discussed in this chapter.

Digital signal processing systems may be operated with different sample rates which makes their data incompatible (it is not provided at exactly the assumed regular times). In particular, A/D and D/A converters can benefit from using sample rates that may not be suitable for the system processing the data (too fast or too slow). To overcome the problem of interfacing data at different sample rates, extra samples may need to be inserted or deleted from the data stream. In either case the spectrum of the output will be corrupted unless appropriate filtering is employed to suppress parts of the spectrum which would be (or become) misplaced. This combination of filtering and sample insertion (or deletion) is called multi-rate filtering.

3.1.2 : Digital Filtering Fundamentals

Many structures exist for implementing a digital filter, the variants offering advantages in terms of required storage space, computation time, or accuracy in converting the intended performance into real performance within a working system. The fundamental structures are those that perform the transfer function in its simplest form and are commonly called the 'direct form' realisations. In most cases this will involve using the minimum number of implementation blocks (adders, multipliers and delays) and this is then a canonic form. Alternative canonic forms that do not use the direct form are used for secondary reasons such as those mentioned above but should still implement the same transfer function.

FIR filters can only implement polynomial transfer functions of the form shown below.

$$TF(z) = \frac{OUTPUT}{INPUT} = \sum_{n=0}^{N-1} c_n . Z^{-(n+1)}$$

Equation 3.1.2.a

The terms in Z^{-n} can be implemented by registers which simply delay the digital signal until the next clock cycle, hence the direct form of this transfer function is as shown below:

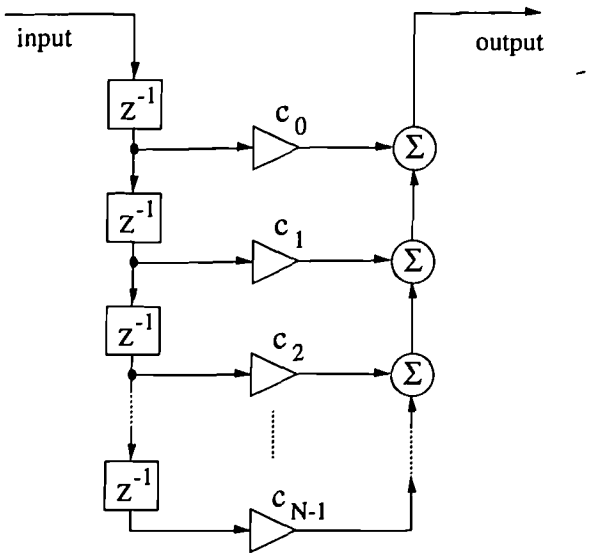


Figure 3.1.2.a : Direct Form Implementation of an FIR Filter

It is worth noting that 'N', the number of coefficients in the filter, sometimes called the number of taps † or the length of the filter, is one greater than the 'order' of the filter which is the same as the order of the polynomial on which the filter is based.

If a digital impulse (a signal of value 1 for one sample and zero thereafter) is applied to the filter the output can be measured for the filter's impulse response. After Fourier transforming, the output sequence can be assessed in the frequency domain yielding its frequency response (an impulse has a white spectrum so after filtering the spectrum follows the shape of the filter's response). Since for the FIR case, the impulse passing through the delay elements in turn causes the output value to be the coefficient value which the impulse is currently aligned with, the impulse response of an FIR filter has the same value as its coefficients. This is particularly useful because a Fourier transform of the coefficient sequence will give the frequency response of the filter.

IIR filters can implement the more general transfer function of the form shown below :

$$TF(z) = \frac{\sum_{n=0}^{N-1}\{c_n . z^{-n}\}}{\sum_{n=0}^{N-1}\{d_n . z^{-n}\}}$$

Equation 3.1.2.b

Under close scrutiny, this transfer function can be seen to be the product of an FIR transfer function (the numerator) and another polynomial term which is in fact the recursive part (the denominator). It is this recursive part which makes the IIR filter very useful, since the signal will pass through the multipliers and adders within its implementation over and over again, making their use very efficient. It is also the cause of instability in the system if not properly designed; when the denominator becomes less than one the signal will grow instead of decaying. Designing IIR coefficients is complicated by this additional stability constraint such that the roots of the denominator polynomial (the transfer function 'poles') have to be forced to be less than one in magnitude. Finding the roots

† The name 'taps' for the coefficients of a digital filter arises from the structure shown on the left of the filter in figure 3.1.2.a, which was first used as a replacement for analogue electronics' "tapped delay line".

requires decomposition which is sensitive to numerical inaccuracies, and implementing the filter in direct form can lead to round-off error which can destabilise the filter (by effectively altering the poles) so IIR filters are commonly designed in quadratic sections, cascaded together. These sections are called 'Biquad' sections and a block diagram of one implementation is shown below :

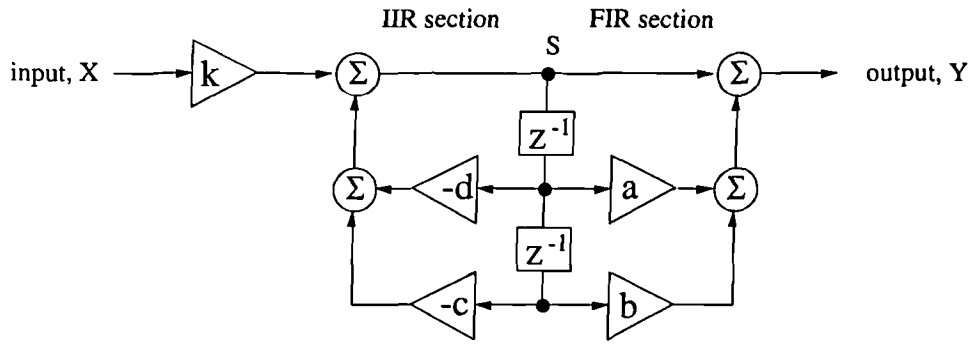


Figure 3.1.2.b : Biquad Structure for IIR Filtering

Using the nomenclature shown above, the transfer function of this section can be found easily :

$$S = kX - S.(dz^{-1}+cz^{-2}) \quad \Rightarrow \quad S = \frac{kX}{1+dz^{-1}+cz^{-2}}$$

and $$Y = S.(1+az^{-1}+bz^{-2})$$

hence $$\Rightarrow \frac{Y}{X} = k \,.\, \frac{1+az^{-1}+bz^{-2}}{1+dz^{-1}+cz^{-2}}$$ Equation 3.1.2.c

IIR filters do not naturally exhibit equal delay to different frequency components (ie. they do not have transfer functions with linear phase). This can be achieved by adding an all pass network after the basic IIR to reverse its non-constant 'group delay' but this cancels the efficiency improvement IIR filters have over FIR filters. To overcome this problem, optimisation routines searching for both specified frequency response and group delay are available; these can produce filters with *approximately* linear phase response. Although IIR filters can be used in most instances of multi-rate filtering, wherever strict linear phase is required (eg. in phase measurements) FIR designs have to be used. In audio applications, non-linear phase filters (ie. with non-flat group delay) introduce effects similar to path length changes between the signal source and the listener. If large this blurs the placement of signal sources as perceived from a stereo reproduction and hence spoils high quality reproduction.

3.1.3 : Sample Rate Increase and Decrease

When a signal is sampled, frequencies near the sampling rate cannot be distinguished from those near DC through a stroboscopic effect. To maintain distinction between frequencies being represented in a sampled system, the sampling rate must be at least twice the highest frequency that is to be represented (Nyquist's sampling theorem for complete signal recovery). To ensure this is the case,

the signal to be sampled must be band-limited before sampling (ie. an 'anti-aliasing' filter should be applied). Commonly, this suppresses the spectrum of the signal outside the range DC to Fs/2 (and for mathematical symmetry, from DC to -Fs/2). In the process of sampling, this portion of the signal is duplicated periodically about multiples of the sampling rate to produce the spectrum shown below :

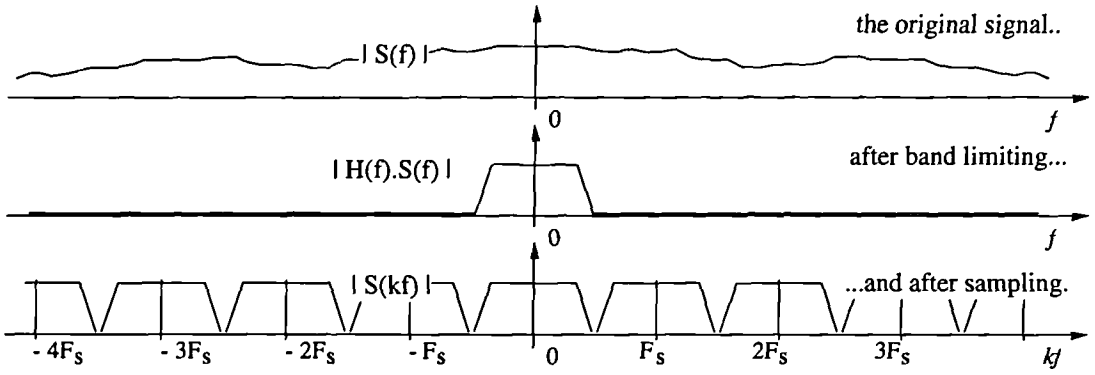


Figure 3.1.3.a : Signal spectrum before and after Band-limiting and Sampling.

Since the digital signal is now only a set of samples (and zero in-between), if additional samples are inserted between and they are set to zero the nature of signal content will not have been changed but its sampling rate will have increased. For instance, if between every alternate sample another is inserted, the sample rate will have doubled (but the spectrum remains unchanged). The spectral replica produced by the original sampling still exists and should be removed by low pass filtering to return the signal to a fair representation of the original analogue signal (as though it had been sampled at the faster rate in the first place). The filter required to do this must pass the baseband signal, and suppress the spectral replicate. Such a filter is called an anti-imaging filter, and when used with sample insertion, the overall process is called interpolation. The filter is often called an interpolating filter since as the name suggests, this process is equivalent to polynomial interpolation as seen in the time domain. These spectral modifications are shown below :

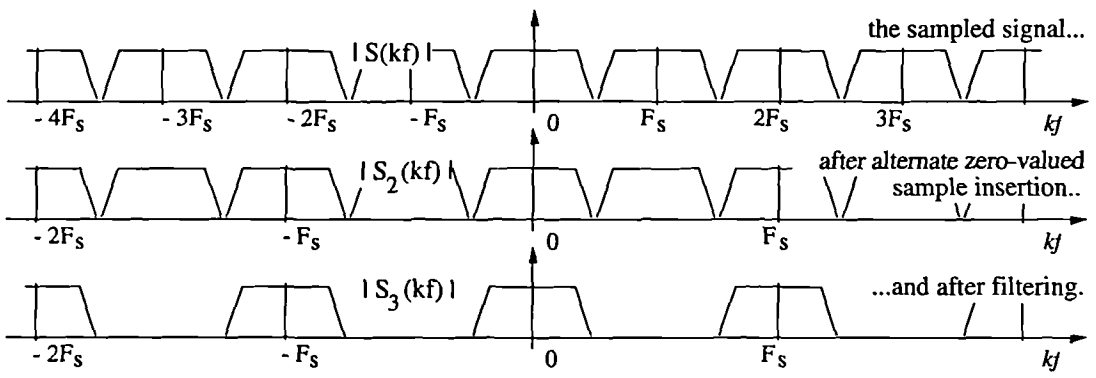


Figure 3.1.3.b : Sampled Spectrum Before and After Alternate Sample Insertion and Filtering

Similarly, if a high sample rate signal is to have samples deleted from the data stream to reduce its sample rate, filtering has to be performed to suppress any spectral components that cannot be represented properly in the low sample rate data (like the band limiting used for analogue signals). Bands where the low rate spectral replicates will be, must be cleared first to avoid their images

corrupting the signal band when the sample rate is reduced. Filters to suppress these components are often called anti-aliasing filters (or decimating filters) and have the same frequency requirements as the anti-imaging filter for the opposite sample rate change. The process of anti-alias filtering followed by sample deleting is called decimation †, and is the dual of interpolation (ie. they complement each other). Note, in interpolation the filtering occurs after the sample rate increase, in decimation the filtering occurs before the sample rate decrease. Thus, interpolating filters generally have a passband gain equal to the interpolation factor (to account for the lack of signal at the zero valued samples) whereas the decimating filters usually have a passband gain of unity.

If a non-integeric ratio of sample rates is required, both interpolation and decimation can be used via an intermediate sample rate used only within the sample rate change block. This is an example of the more general process of multi-stage filtering which will be returned to in section 3.2.3. This idea will be returned to in chapters 6 and 7 where fine-tuning of the sample rate used in a practical implementation is explained using asynchronous sample rate conversion (ASRC).

Four main areas exist in the design of PWM DACs which require the use of multi-rate filtering techniques. These are output decimation for feedback or analysis, and input interpolation (required for recovery filter simplification, for PWM linearisation, and for noise shaper operation). These applications are discussed in brief below and then decimation for analysis will be discussed again in more detail in section 3.4 .

3.1.4 : Applications of Multi-rate Filtering in PWM DAC Design

The first application of multi-rate filtering that will be looked at is that of spectral analysis of the output of a DPWM. This is used to allow precise comparison of the performance of alternative PWM DAC designs with each other and PAM and ΣΔ designs. When data is D/A converted using DPWM, the output can be viewed as a single bit digital stream clocked at the product of the input sample rate and the number of input levels. This output stream can be spectrally analysed using an FFT (after windowing) but due to the high sample rate of the output, detail in the spectrum can only by observed if very large FFTs are employed. This involves applying a very high order FFT (typically 10^6 points) which can be prohibitively slow, uses vast amounts of storage space (c. 32 MBytes) and will produce considerable unwanted output if only a small section of the spectrum is required in great detail (eg. the audio band or the near-carrier bands).

Special 'zoom' FFTs can be applied to speed up this process by precalculating much of the FFT beforehand and knowing the structure of the PWM signal (evaluating only high periods, using addition since the pulse amplitude is unity, etc.).

Fast analysis can be performed in some cases by reducing the sample rate first (by decimation) and then applying a shorter FFT solely to the band of interest. In this case most of the processing effort is used in performing the decimation, but this takes less time than either the complete or zoom FFT approaches since the quantity of data to process is repeatedly reduced in this technique. Furthermore, the processing can be simplified by taking advantage of the structure of the PWM signal, using precalculated filter output values, applying transfer functions with zero coefficients (which do not

† The term *'Decimation'* is derived from the Roman army's threat to kill one in *ten* of the surviving soldiers if a legion lost a battle; in DSP it is used for sample rate reduction of any factor.

have to be evaluated) and using integer mathematics in the filtering (as far as possible). An example of this is discussed in section 3.4 .

The second application of multi-rate filtering that will be looked at is that of producing a band-limited version of the DPWM output. This may be required when the output is used elsewhere in the system; for example, as a feedback signal in a closed loop PWM DAC (see chapter 5). As above, the DPWM output is treated as a high-speed digital signal, with its sample rate at some product of the input sample rate, and then decimated to produce the lower rate version.

In contrast with bitstream DACs, PWM DACs are usually operated with very high output sampling rates, so 'multi-stage', multi-rate processing is ideal for this application. The reason for such high sample rates is as follows: PWM is preferred as a D/A conversion technique since it reduces the number of transitions per second in the output signal (this reduces wasted power, allows power switch implementation, etc.); this implies using the *lowest* oversampling that can be tolerated. To maintain high resolution, the *largest* data wordlength that can be tolerated has to be used in the modulator. This implies that the output sample rate is the *largest* multiple of the output pulse repetition rate that can be tolerated, typically 256x or more. For example, with an input oversampled by at least five times the Nyquist rate, and a data wordlength of at least 9 bits, the output sample rate is more than 2500 times that required for the signal.

In multi-stage decimation the initial sample rate change can be performed in large ratios, allowing aliasing to occur in bands that will subsequently be suppressed. This simplifies the filtering required by relaxing the initial filter requirements (see section 3.2.3). For example, ratios of 18, 8, and 2 are used for reduction by a factor of 288 overall in a case study in section 3.4; such filtering can also be implemented using special structures (see section 3.4.2 & 3) and using easily implemented transfer functions (see section 3.2.4, 5 & 6).

The third application of multi-rate filtering is that of input signal interpolation to imitate oversampling. This can be used to simplify the requirements of an output recovery filter provided the D/A converter can handle the increased sample rate implied by this operation. This technique is not unique to PWM DACs since output recovery filters are required for most systems to allow linear behaviour of subsequent amplifier stages and avoid overload of loudspeakers with sampling frequency power. Signal interpolation also permits the use of simpler noise shaping, a point which will be returned to in chapter 4.

In order to recover the analogue signal from the output of any DAC, analogue low pass filtering is used to remove spectral replicates of the signal which were introduced by the original sampling of the signal. If this filtering is required to pass the first spectral lobe (0-20 kHz) and suppress all others (24.1 kHz onwards) to the resolution of the system (≈100 dB for 16 bit resolution) a very high order filter is required because of the high rate of change of the required amplitude response directly after the passband. For example, taking a general passive, analogue, low pass filter with a roll-of rate approaching 20 dB per decade per order, and a transition band of 4.1 kHz after 20 kHz (0.081 of a decade) to suppress to ≥ 100 dB, a filter of at least 62 $^{nd.}$ order would be required. Implementation difficulties prevent such filters from being constructed, so lower order filters are used, allowing small spectral replicates in the output. Unfortunately, to reduce the spectral replicates to a sufficiently low level, some passband amplitude roll-off has to be tolerated, and significant passband phase distortion

† The sample rate here is equal to the modulators internal counters' rate. This should not be confused with the pulse repetition frequency which will be considerably lower for PWM DACs than bitstream DACs.

can be introduced.

Recovery filter shortfalls can be avoided if the nearest spectral replicate is at a higher frequency, ie. the signal is sampled at a higher rate. This allows a far wider transition band to be assumed which greatly reduces the complexity of the filter required to fulfil the requirements. Storage or transmission of the signal sampled at a higher rate is wasteful, providing excessive channel redundancy for complete signal recovery, so digital sample rate change is preferred between decoding the stored or transmitted data and the D/A conversion stage. Interpolation by a factor of four or eight is common because DACs capable of handling input data at these rates are not significantly more expensive to produce than ones operating at or near the Nyquist rate for audio (c. 40 kHz). Assuming a Butterworth low pass filter is used with a 50 kHz passband (Butterworth filters exhibit near linear phase performance for the first 40% of the passband), and a sample rate of 352800 Hz (8x interpolation), the transition band is increased to ≈0.75 decades, and a seventh order filter will reduce the spectral replicates to ≥ 105 dB below the signal. Less aggressive filtering may well be adequate and is largely dependent on the circuitry which follows. More linear phase performance can be found in Bessel (Thomson) filter designs but the benefits of this are largely subjective since the degree of spatial blurring by slightly non-linear phase designs is small.

The fourth application of multi-rate filtering is that of interpolating the signal to permit more linear performance from the PWM stage of the converter. There are three published ways in which this can be used to advantage. Firstly, as shown in chapter two, the PWM distorts signals increasingly as the signal-to-carrier frequency ratio is increased. To avoid the PWM operating in such regions oversampled input can be used [SAN83]. For simple modulation types, the degree of oversampling required is enormous, prohibiting practical modulator design for complete linearisation, but with appropriate choice of modulation type, low distortion performance for signals at common amplitudes can be achieved in this way [HIO91a]. By using more advanced modulation techniques such as 2SCPWM in combination with moderate oversampling (PRR≈8x Nyquist sampling rate) very high linearity can be achieved. This requires oversampling by at least 4x and becomes useful up to about 16x which can easily be imitated by interpolation. For such systems, the oversampling ratio required is controlled more by the noise shaper design, the output recovery filter complexity, and in the limit - the separation of the signal from sideband terms of intermodulation between signal and carrier. The second use is in combination with a pre-compensation technique to imitate natural sampling as shown in figure 5.2.1.b. The third published use is in the refinement [PED94] of a technique [MEL91, LEI90A, CHE92] to approximate natural sampling. Discussion of these will be left until chapter 5.

It should be remembered that oversampling can only be imitated once the signal has been sampled and quantised since the quantisation noise added at this stage is spread over the baseband and will not be removed by the filtering used in interpolation to suppress out-of-band features (spectral replicates). Because of this the SNR of an interpolated signal cannot improve on that of the original sampled signal in the way that oversampling an analogue signal can increase the narrowband SNR by 3 dB per octave of oversampling. It is also worth remembering that while even naturally sampled PWM types require some oversampling to avoid sidebands interfering with the baseband, the uniformly sampled PWM types require only a little more to be effective for normal signal levels [HIO91a].

3.2 : Improving Multi-rate Filtering Techniques.

3.2.1 : Overview

While operating filters for sample rate conversion several adjustments can be made to reduce the amount of computation required. Firstly, the data structure produced by sample insertion or deletion can be taken advantage of, reducing the filtering process by the interpolation (or decimation) factor. Secondly, for sample rate change by ratios which are not prime, the sample rate change can be done in multiple stages, allowing relaxed filter requirements for most of the filters and hence reduced order and complexity. Thirdly, special filter impulse responses can be used with alternate coefficients of zero that do not need to be evaluated or coefficients of simple binary numbers that can be evaluated with minimal effort. Fourthly, in FIR filters with symmetry in their impulse response (ie. linear phase behaviour), an additional factor of two can be achieved by reordering the evaluation of the filter.

3.2.2 : Reducing Computation by the Interpolation (or Decimation) Factor

When FIR filters are used in sample rate reduction, substantial reductions in the computational complexity can be achieved by only evaluating the required output samples (similarly in sample rate increase only the non-zero samples applied to the anti-imaging filter are required). This does not imply that the filtering and sample deletion/insertion can be commuted since samples cannot be dropped before filtering in decimation (or assumed to be zero after filtering in the interpolation case). Rather, the signal applied to the decimation filter can be stepped in increments of the decimation ratio and the filter used in the interpolator will only have even indexed coefficients that are non-zero. The transfer function for this is shown below :

$$TF = \sum_{k=0}^{M.(N-1)} a(k).z^{-kM}$$

Equation 3.2.2.a

This implies that the computation rate is equivalent to a filter operated (convolved) at the low rate side of the decimator or interpolator. For this reason, narrowband filtering even without sample rate change is sometimes operated using a multi-rate scheme (see section 3.2.3 and 3.4.6).

A similar reduction in the computational complexity can be achieved in the IIR case by removing all but the coefficients corresponding to the M^{th} outputs (that is coefficients for the powers of z^{kM}. For this to be used the required transfer function has to be re-expressed as :

$$TF = \frac{\sum_{k=0}^{L} a(k).z^{-kM}}{1+\sum_{k=1}^{K} b(k).z^{-kM}}$$

Equation 3.2.2.b

Designing for such transfer functions is not simple since the closed form equations for the classical designs (Butterworth, Chebychev I & II, Thomson, Elliptic etc.) cannot easily be transformed to suit this transfer function. Design by optimisation can still be applied with revised target functions and this is described in section 3.3.5.

3.2.3 : Multi-stage Filtering Requirements

When high decimation or interpolation ratios need to be achieved, the band that has to be protected from aliasing or imaging is very small compared to the sample rate, thus the transition band between the passband and stopband is also small which forces a large order filter to be used to meet the filtering requirements. This suggests that a large amount of data will be processed with a large filter and the computational overheads are enormous. If the task can be split into parts by factorising the overall ratio into smaller ratios and running a sequence of sample rate conversion stages, some of the stages will have a reduced amount of data to process and some of the stages will have a reduced complexity of filtering to apply (knowing that other stages can be relied on to filter out certain bits of the spectral replicates). This leads to great savings in the amount of computation required, so much so that simple filtering without sample rate change is often done in this way.

In interpolation, the simpler filter characteristic arises because the initial stages clear a portion of the band which can then be relied on to be empty (and hence cannot introduce further images even if moved by sample rate change). In the decimation case, this same band can be aliased into, knowing that the final stages will filter out the allowed aliasing before using it. Thus, except for the lowest rate stage, the transition band of each filter can be made wider even if this would allow imaging or aliasing if the filter was used on its own. It should also be remembered that this filter can no longer be used in isolation since it is the 'teamwork' of successive filtering which meets the system requirements. As before, each stopband has to sufficiently suppress unwanted spectral components to ensure they are insignificant, although in critical designs the accumulated effects of several filters can be allowed for to slightly ease transition band requirements (this effect is very small).

When using multiple stages, the passband ripple of successive stages can begin to accumulate and these ripples can constructively form an overall performance that no longer meets the system requirements. In order to combat this, the passband has to be equalised or the individual passband specifications have to be tightened by a factor of the number of stages required. Fortunately, some useful filter designs exhibit passband over-design to allow impulse responses which are easier to implement (see section 3.2.4) so tightening the passband is not a severe penalty to pay.

An example set of filters for decimation by ten is shown below with the wider transition bands marked as Φ.

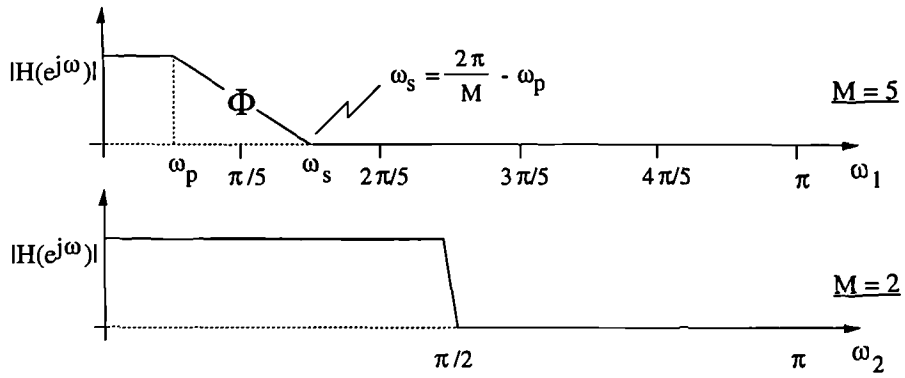


Figure 3.2.3.a : "Don't Care" Band in an 10x Multi-stage Decimator.

3.2.4 : Half-Band Filters

Half-band filters have the property of odd symmetry in the frequency response about $F_s/4$ ($\omega=\pi/2$), which implies that the filter must have alternate even indexed coefficients equal to zero (and thus need not be evaluated), except k=0 . ie.:

$$h(k)=\begin{cases} 1, & k=0, \\ 0, & k=\pm 2,\pm 4,\ldots \end{cases} \qquad \text{Equation 3.2.4.a}$$

Symmetry about $\omega=\pi/2$ implies that the stopband and passband are of the same width so these filters can only be used for a decimation or interpolation ratio of two. Odd symmetry in the frequency response forces the magnitude response at $F_s/4$ to equal 0.5 and hence some aliasing near the mid-band frequencies must be tolerated if a half-band filter used. Furthermore, the passband and stopband ripple must be the same size which for filters for high resolution systems forces the passband to be over-designed since these use very small values of stopband ripple. Although this means that the half band design is sub-optimal for satisfying the design requirements, this does not take into consideration the reduction in computation through not evaluating zero-valued coefficients. Fortunately, the order required for given passband and stopband characteristics (equation 3.3.2.c) is not very sensitive to passband ripple so this over-design only increases the filter order slightly and permits a computational saving approaching 50%.

One particular family of half-band filters is useful for the additional simplicity of their integer coefficients [GOO77]. Shown below is a table of these filters, which when used in conjunction with comb filtering (see section 3.2.6) can be used to dramatically reduce the computation required for many interpolation/decimation applications involving factors of two. Three of these filters are monotonic (F2,F3 & F5) and the remainder are equi-ripple as shown in figures 3.2.4.b & c which are the linear and logarithmic magnitude, frequency responses.

Filter No.	Filter Delay	Passband Gain	0	±1	±3	±5	±7	±9
F2	2	4	2	1				
F3	3	36	16	9	-1			
F4	3	64	32	19	-3			
F5	4	512	256	150	-25	3		
F6	4	692	346	208	-44	9		
F7	4	1024	512	302	-53	7		
F8	5	1604	802	490	-116	33	-6	
F9	6	16384	8192	5042	-1277	429	-116	18

Figure 3.2.4.a : A Table of Useful Integer Coefficient Half Band FIR Filters.

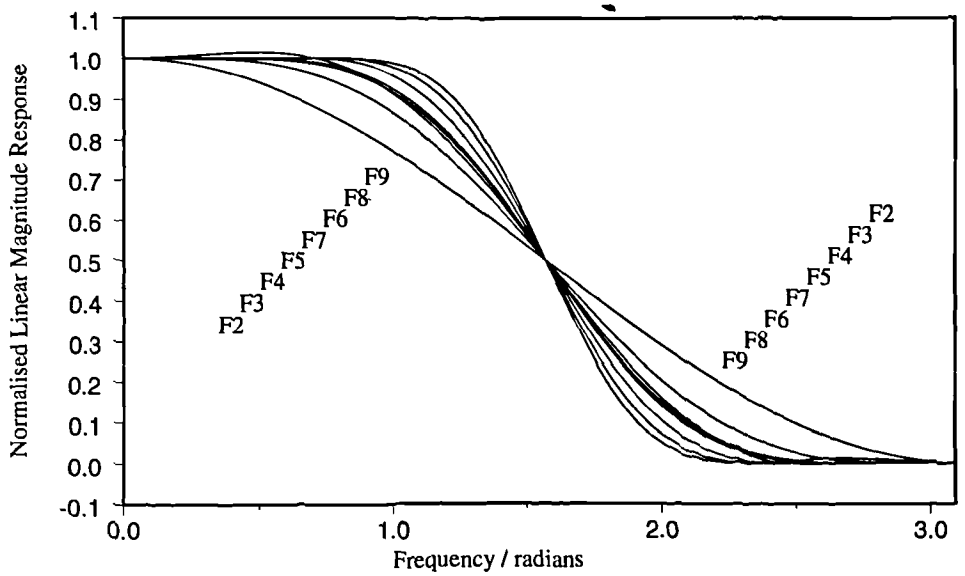


Figure 3.2.4.b : The Linear Magnitude, Frequency Response of a Family of Half Band FIR Filters.

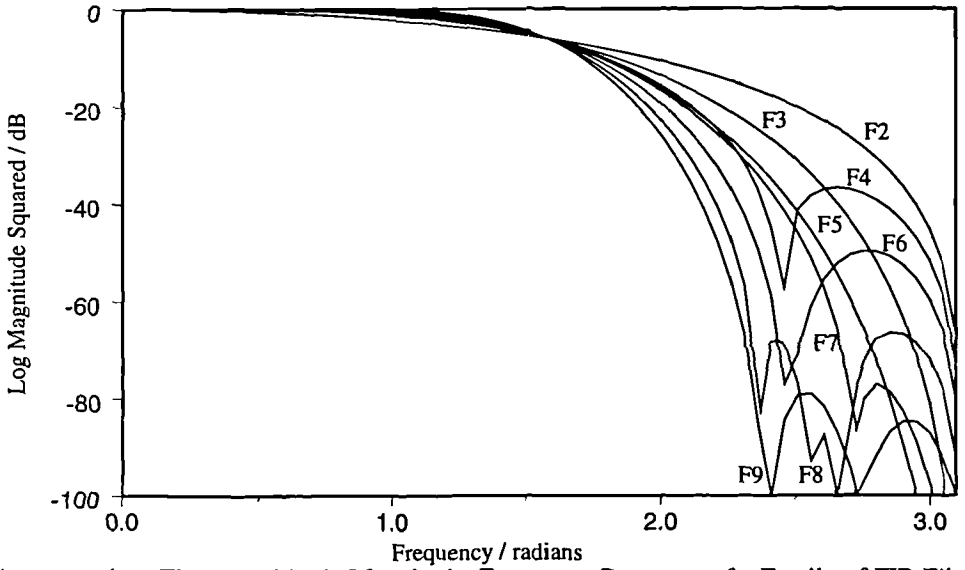


Figure 3.2.4.c : The Logarithmic Magnitude, Frequency Response of a Family of FIR Filters.

3.2.5 : Multi-Band Filters

Multi-stage design using a sequence of decimators can be made very efficient since each stage provides a good compromise between passing-on less data and using a higher order. Ratios which are powers of two can be handled particularly efficiently using a chain of half-band filters (see section 3.2.4) but stages with prime ratios greater than two will have correspondingly smaller transition bands (which forces the use of higher order filters).

Using purely low pass filters adds unnecessary complexity to the processing chain since, except for the lowest rate stage or ratios of less than three, there exist multiple bands that can be used for aliasing into which will still be filtered out by lower rate stages. Filter order can be reduced by

allowing use of these 'don't care' regions or 'Φ'-bands and in this way allow lower computational effort. Filters designed to take advantage of this additional degree of freedom are called multi-band filters. An example of the filter frequency response for decimation by ten is shown below :

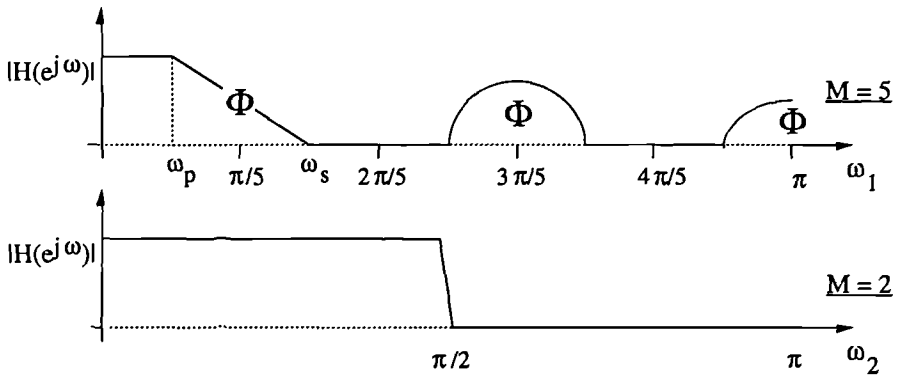


Figure 3.2.5.a : Multi-stage Decimation by Multi-band Filter - Required Frequency Responses.

This multi-band design can be directly compared with the low pass multi-stage design as shown in figure 3.2.5.a, noting the transition bandwidth of the passband remains the same, multiple stopbands have been defined :

$$\omega_{s,k} \leq K.\omega_{in}/M \pm \omega_p \text{ , } K<M/2 \qquad \text{Equation 3.2.5.a}$$

and that the additional aliasing bands have been allowed to co-exist knowing that these will be suppressed in the lower rate stage.

In the high sample rate stages, multi-band designs can be used without incurring much additional processing even for non-prime ratios (compared to half-band implementations) and enabling a significant reduction in the processing required compared to the purely low pass design.

The benefits of this are most pronounced if the bandwidth of the 'don't care' bands is large as a proportion of the half sample rate although the special properties of half-band filters (alternate zero coefficients) are lost. In the limit of large 'Φ'-bands, multi-band filters tend towards 'comb' filters which will be discussed in the next section.

3.2.6 : Comb Filters

Comb filters are based on an impulse response which is an impulse train of length, 'N', and hence have a coefficient set with all coefficients equal to one as shown below for a sixteen tap filter.

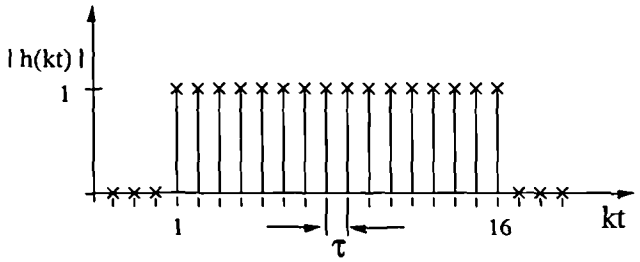


Figure 3.2.6.a : Impulse Response for a Sixteen Tap Comb Filter.

These filters are particularly simple to implement since they do not require any multiplication (other than scaling) which makes them particularly useful in simulations where a large amount of data has to be handled or in real time processing where fast filtering is important.

Comb filters have N-1 highly selective stop bands, equi-spaced in frequency for $0 < \omega < 2\pi$. These correspond to the N-1 zeros of the filter's transfer function which are equi-spaced in angle and on the unit circle in the complex-Z plane. From this the filter family can be seen to have both linear phase response and minimum phase lag. The frequency response and pole-zero plots are shown below for the 16 tap case as used above. It should be noted that the passband response of the comb filter is not flat (although it approaches a flat response at DC) and hence passband compensation may be required in subsequent filtering to correct for this.

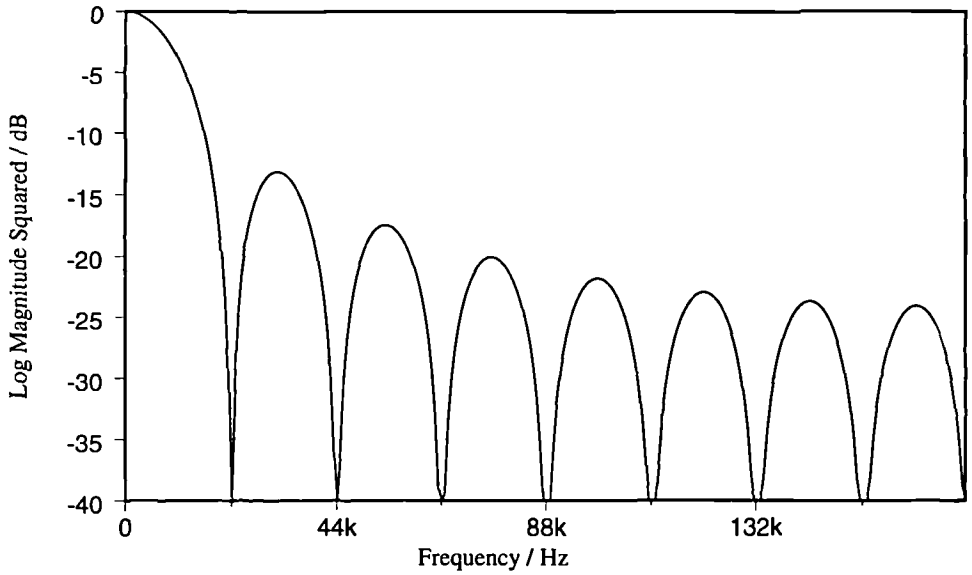


Figure 3.2.6.b : Frequency Response of a Sixteen Tap Comb Filter.

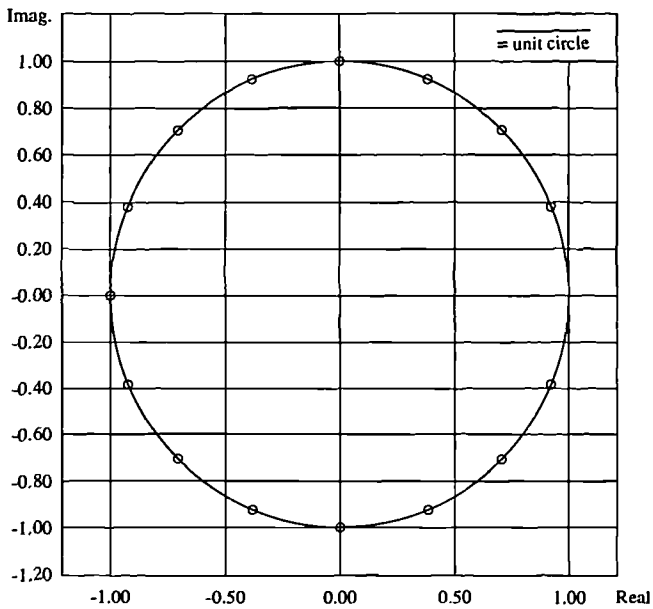


Figure 3.2.6.c : Organisation of Zeros on the Complex-Z Plane For a Sixteen tap Comb Filter

Since the stopbands of comb filters are so narrow, these are only useful in high resolution systems if used in the early stages of high ratio decimation (or the final stages of high ratio interpolation). For applications where lower sample rate change ratios are to be employed at high resolution (eg. 2048x at 16 bit resolution), a cascade of similar comb filters may be used. This increases the suppression (in dBs) by a factor of the number of cascaded combs used since the zeros coincide. The magnitude of the Fourier transform of the impulse response of an N-tap comb filter is :

$$|H(e^{j\omega})| = \left|\frac{Sin(\omega . N/2)}{Sin(\omega/2)}\right| \qquad \text{Equation 3.2.6.a}$$

so a 'P'-stage cascaded version exhibits a suppression, 'S', of :

$$S = 20 . P \, Log_{10} \left|\frac{Sin(\omega . N/2)}{Sin(\omega/2)}\right| \qquad \text{Equation 3.2.6.b}$$

Such compound comb filters will be referred to as second, third or fourth order comb filters (ie. for P=2,3,4) which should not be confused with the number of taps found within each filter. Since comb filters are a type of multi-band filter these are particularly suitable for decimation and interpolation by a factor of N, and can easily be implemented by an 'accumulate and output' strategy. From equation 3.2.6.b, the achievable suppression for the required filter can be found and the passband loss can be assessed for its significance (and whether or not it is worth compensating for in subsequent stages). The stopband suppression is generally the more difficult parameter to satisfy and takes its worst value (produces the highest aliasing) from the lower edge of the first stopband. A rough estimate of the suppression can be found for high ratios, since :

$$S \approx 20 . P . Log_{10} \left|\frac{\text{eventual passband width}}{\text{high sample rate}}\right| \qquad \text{Equation 3.2.6.c}$$

Comb filters can also be implemented recursively using the same impulse response, but a different evaluation architecture. In each sample period the output differs from its predecessor by the additional (current) input value and the removed input value from N sample periods previously. Thus the comb filter can be implemented as an accumulator (1 st. order digital integrator) and an FIR filter with transfer function : $H(z) = 1-z^{-N}$ (a 1 st. order digital differencer). A block diagram of this is shown below :

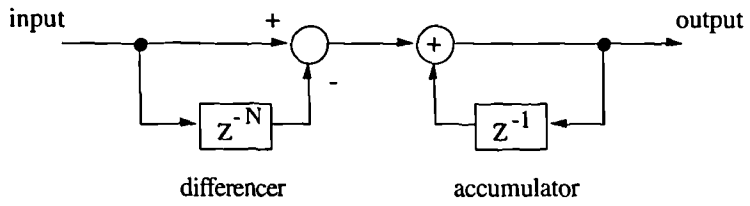


Figure 3.2.6.d : The Recursive Implementation of a 1 st. Order Comb Filter

Higher order comb filters can be implemented by cascading the above structure but care is required in the evaluation of the accumulator since this will accumulate calculation error as well as the signal. DC content in the signal is also of concern since this can rapidly cause overflow unless the signal has been differenced more times than accumulated before each accumulator block in the

processing chain. One elegant solution to dynamic range limitations is to use fixed point signal representation and permit overflow, modulo 2^{B-1} (where B is the number of bits in that part of the processing chain); the paired accumulator or differencer will then restore the signal by under-flowing before the output. In this application, integer signal representation is better than floating point, since it prevents limited wordlength in the accumulator introducing round-off error (which is likely due to extreme dynamic range requirements at this point). If evaluation is carried out in this way, all the accumulators can be placed before the differencers which permits yet further architecture simplification:

Where a recursive comb is used in a multi-rate application, only 1/N of the output samples are required, but normally the recursive implementation of a comb filter would require that the differencing and accumulating is operated for each of the input samples. Since the differenced output depends on only the current input and the input N sample periods before, and the sample rate reduction will discard the N-1 samples between, the differencer part of the comb filter can be placed after the sample rate conversion [SAR90]. In this case, the computational complexity of the whole filter almost halves to just P.(N+1) additions per output sample, as a result of running the differencing part of the transfer function (the numerator, ie. FIR, part) at the output sample rate as explained in section 3.2.2. This structure for a second order comb in a multi-rate system is shown below :

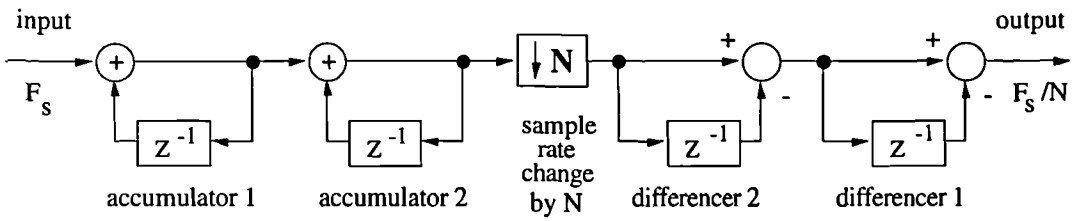


Figure 3.2.6.e : A Second Order Recursive Comb Filter With Differencers After Decimation.

3.2.7 : Computation Reduction by using FIR Symmetry

Linear phase FIR filters exhibit symmetry in the impulse response which enables grouping of the multiplication operations carried out in the process of operating the filter (convolving it with the signal). Summation in simulation and hardware is always simpler and usually faster then multiplication except for when the multiplication can be achieved by simple shift operations. By factorising the convolution to be applied, as below, summation can be performed before multiplying with a resulting saving in computation of a factor of two for the linear phase FIR case.

$$TF = a_{(N+1)/2} \cdot z^{-(N+1)/2} + \sum_{n=0}^{(N-1)/2} \left\{ [z^{-n} + z^{-(N-n)}] . a_n \right\} \quad \forall \text{ N, odd.}$$

Equation 3.2.7.a

A modified structure is used to implement this filtering as shown overleaf.

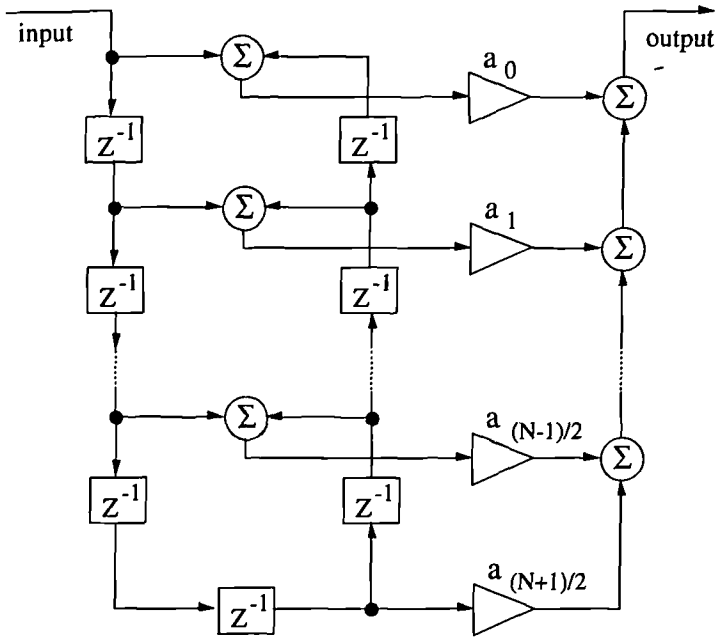


Figure 3.2.7.a : Modified FIR Structure to take Advantage of Symmetry in the Impulse Response. †

Taking advantge of the symmetry of linear phase filters reduces the multiplication content of the complexity of the filter by a factor of two but in DSP chips where dedicated hardware where multiplication is performed in the same time as addition this saving becomes only a 25% saving. This technique also prevents other the fast techniques being used such as fast FIR convolution [CHE92] (see section 3.4.5) although it offers comparable savings. Where linear phase is not a requirement in the filtering used (eg. where only the final magnitude spectrum is required) shorter filters can be found which do not exhibit linear phase (eg. minimum phase filters) and in this case the alternative convolution speed up would be favoured.

† this example is for an odd order filter; in the even order case, the bottom delay is replaced with a summing node as used higher up, and the bottom coefficient index is simply N/2.

3.3 : Filter Design Techniques.

3.3.1 : Overview

Each stage in a multi-stage, multi-rate filtering application uses a filter with individual requirements. In the design procedure, the required order should be specified; for this an estimate of the order to satisfy the other requirements has to be made. If linear phase is not a requirement, IIR filtering can be chosen for all but the simplest of stages where simple coefficients may make an FIR filter a viable alternative. In many instances the IIR filter can be modelled by an analogue filter for which design equations and the required order estimators are available. This can then be transformed into the biquad coefficients directly or converted to poles and zeros of the transfer function and optimised to find better coefficients to satisfy approximate linear phase performance.

If FIR filtering is to be used, the order can be estimated from empirically derived tables or using the equations shown in the next section, and then a suitable impulse response can be found from which the coefficients can be taken. Two approaches to filter design are commonly used, firstly defining an ideal low pass filter and minimising the squared error, and secondly minimising the maximum error of the frequency response (minimax, or equi-ripple design). Minimising the squared error spreads the time domain error evenly so that the error variance is smaller than in the minimax design. Input sample values do not appear at the output in least-squared designs and have to be calculated so minimax design offers computational advantages in implementation (by not calculating output samples that exist in the input). For this reason, least squared design is not considered further.

3.3.2 : Order Estimation for Low Pass, Linear Phase, Equi-ripple FIR Filters

The application that the multi-rate filter is required for, will normally place constraints on the sample rate ratio between the output and the input, the flatness of the frequency response and the linearity of the filter phase response. Also, the system resolution will set a minimum requirement on the amplitude of spectral replicates. These can be converted into passband and stopband requirements in amplitude and frequency which can be plotted as shown below:

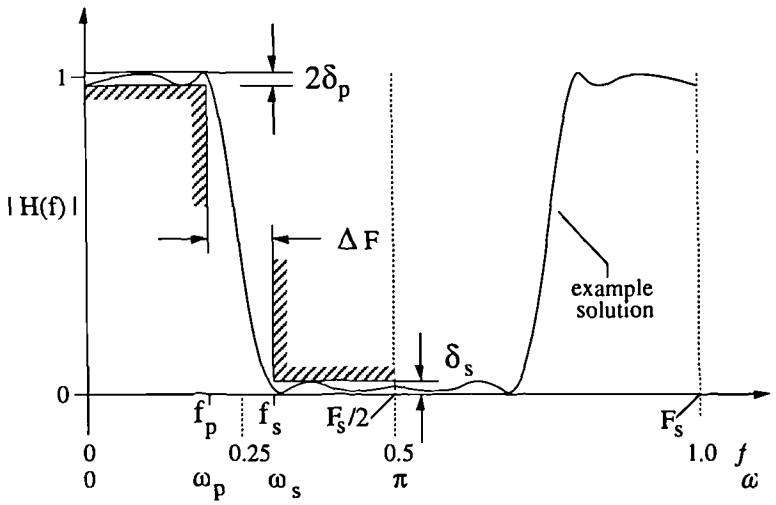


Figure 3.3.2.a : Interpolating (Decimating) Requirements for 2x Sample Rate Change Filter.

From the specified maximum amplitude deviation ('ripple') for the passband (δ_p) and stopband (δ_s) and the cutoff frequencies for the passband (f_p) and stopband (f_s), the filter can be calculated for the closed-form design formulas (eg. Butterworth, Chebychev, etc.). If required, results from these approximations can then be applied to optimisation routines (see section 3.3.5) to modify the impulse response or simultaneously optimise the delay response as well.

For the linear phase FIR case the minimax error filter can be constructed by a multiple exchange Remez algorithm (as discussed in section 3.3.3) using the above parameters and the required order, 'N'. Empirically derived relationships between these five parameters can be used [CRO83] to estimate any parameter from the other four via the three intermediate quantities as shown below:

$$D = (N-1).\Delta F = \frac{(N-1).(\omega_s - \omega_p)}{2.\pi},$$

$$D_\infty(\delta_p, \delta_s) = \mathrm{Log}_{10}\delta_s.[\,a_1(\mathrm{Log}_{10}\delta_p)^2 + a_2\mathrm{Log}_{10}\delta_p + a_3\,] + [\,a_4(\mathrm{Log}_{10}\delta_p)^2 + a_5\mathrm{Log}_{10}\delta_p + a_6\,],$$

$$f(\delta_p, \delta_s) = 11.012 + 0.512(\mathrm{Log}_{10}\delta_p / \delta_s) \quad \text{for} \quad |\delta_s| \le |\delta_p| \qquad \text{Equation 3.3.2.a}$$

where: $a_1 = 0.005309$, $a_2 = 0.07114$, $a_3 = -0.4761$, $a_4 = -0.00266$, $a_5 = -0.5941$, $a_6 = -0.4278$.

These are related by : $$D_\infty(\delta_p, \delta_s) = D + f(\delta_p, \delta_s).(\Delta F)^2 \qquad \text{Equation 3.3.2.b}$$

From this, the necessary order, 'N', to meet the specifications above can be estimated as shown below and rounded up to the nearest acceptable integer .

$$N = \frac{D_\infty(\delta_p, \delta_s)}{(\omega_s - \omega_p)/2\pi} - f(\delta_p, \delta_s).\frac{\omega_s - \omega_p}{2\pi} + 1 \qquad \text{Equation 3.3.2.c}$$

These equations are programmed into 'order .pas' (in appendix A.3.1) to estimate filter length prior to filter design and provide good estimates for low pass designs (and by symmetry , high pass designs).

3.3.3 : Design of Low-Pass, Linear-Phase, Equi-ripple FIR Filters

In designing single and multi-stage decimators and interpolators, optimal filters are required which meet the system constraints δ_p, δ_s, ω_p, ω_s, for the estimated order, N as predicted by 3.3.2.c . The frequency response of the filter will always be approximate in the sense that at some frequencies it meets the requirements and at others it will better the requirements. Bettering the requirements at any frequency implies an over-design that has to be balanced by a greater computational complexity (typically a higher order), so the optimal filter is that which spreads the approximation error evenly distributing any over-design as thinly as possible.

Iterative procedures have been developed along these lines for designing FIR linear phase filters, fixing either δ_p and δ_s, or ω_p and ω_s . Assuming a central coefficient, h_0, which sets the gain, and 'M' free coefficients in the filter, (M = {N-1}/2), the remaining design variables can be adjusted to construct trial filters. Since the filters have linear phase, the frequency response can be written as :

$$H(e^{j\omega}) = h(0) + \sum_{n=1}^{M} 2h(n).\mathrm{Cos}(\omega n) \qquad \text{Equation 3.3.3.a}$$

Re-writing this response in terms of powers of cosine functions shows that the response is in fact an M^{th} order trigonometric function; taking the derivative of this shows that there must be M-1 maxima and minima and that the function takes on a gradient of zero at $\omega=0$ or π. This is particularly significant since M-1 extrema in an M^{th} order trigonometric function implies that for equi-ripple performance there will be M-1 frequencies at which the frequency response is known ($1 \pm \delta_p$ in the passband or $\pm \delta_s$ in the stopband, a 14^{th} order (15 tap) example is shown below).

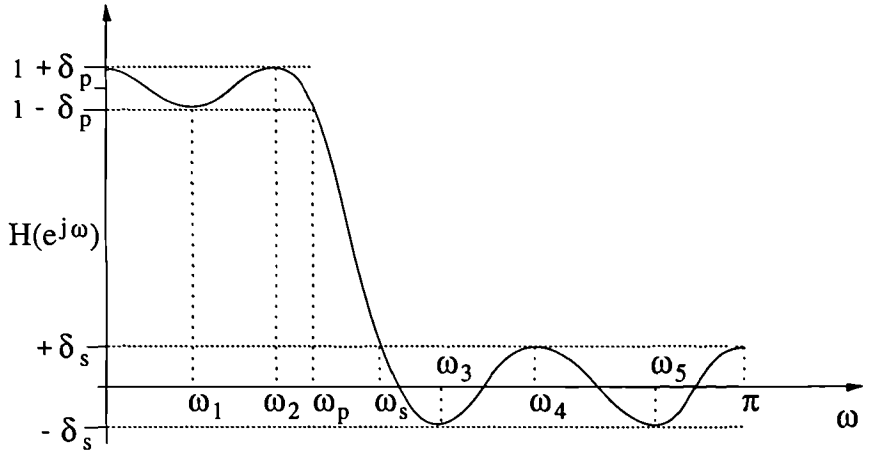


Figure 3.3.3.a : A 14 th. Order Equi-ripple Frequency Response

A set of 2.M non-linear equations, can be used for low values of M (<30) to iteratively solve for the M-1 extrema's frequencies and the M+1 unknown coefficients in the filter (h_0 & M other free coefficients) [HER70]. By altering the share of extrema in the passband and stopband an optimum filter can be found, but owing to the difficulty of solving the non-linear equations this method is not suitable for high order designs.

For designs with order above thirty, an iterative technique can be used, choosing the number of extrema in the passband and stopband, N_s and N_p, estimating the extremal frequencies, and formulating a trial filter by Lagrangian interpolation. In each successive iteration, the extremal frequencies for the next Lagrangian interpolant are taken from the maxima and minima of the current interpolant and their values are set to the expected passband and stopband ripple values. This approach [HOF71] has been shown to converge, but does not allow precise control over the values of ω_p and ω_s. To fix ω_p, ω_s , and approximate δ_p, δ_s , Chebychev approximation over disjoint sets can be used [PAR72].

This problem can be recast, minimising δ_s over the portion of the band $\omega_s \leq \omega \leq \pi$, and δ_p over the band $0 \leq \omega \leq \omega_p$ (for the low pass case); it has been shown that the optimal filter is achieved if the response has at least M+2 "alternations" in these regions (where the error in the response swings from the largest positive to the largest negative, eg. $\pm \delta_s$ in the stopband).

As before, a set of equations can be written and solved, but in this case it need only be done for the value of the of maximum error, ρ, from which a trigonometric polynomial can be formed with error ρ at estimated extrema frequencies. Taking the maxima and minima of this polynomial as the new extrema frequencies their values can be set to $\pm \rho$ in the stopband and $\pm \rho.\delta_p/\delta_s$ in the passband until the equation solution, ρ, does not change appreciably from one iteration to the next. At this point, ρ is the best value that can be achieved for δ_s with a given ω_p and ω_s ; from this, δ_s can be calculated. Once the best share of extrema in the passband and stopband have been found, and the best approximate

response is known, an inverse discrete Fourier transform can be used to find the impulse response which can then be sampled at the required N points to find h(t).

A programme for designing filters based on this technique [MCC73b] has been used for all linear phase FIR designs in this thesis. One example filter's pole-zero plot, impulse, step and frequency responses are shown below.

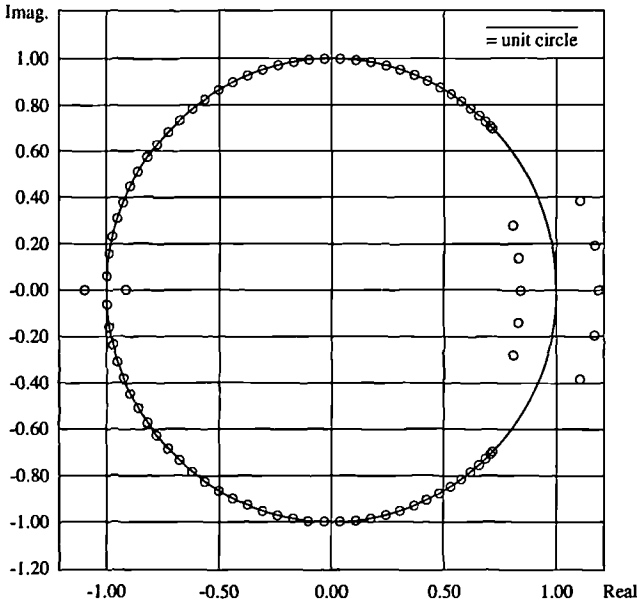


Figure 3.3.3.b : Pole Zero Plot for an Equiripple Filter FIR Filter.

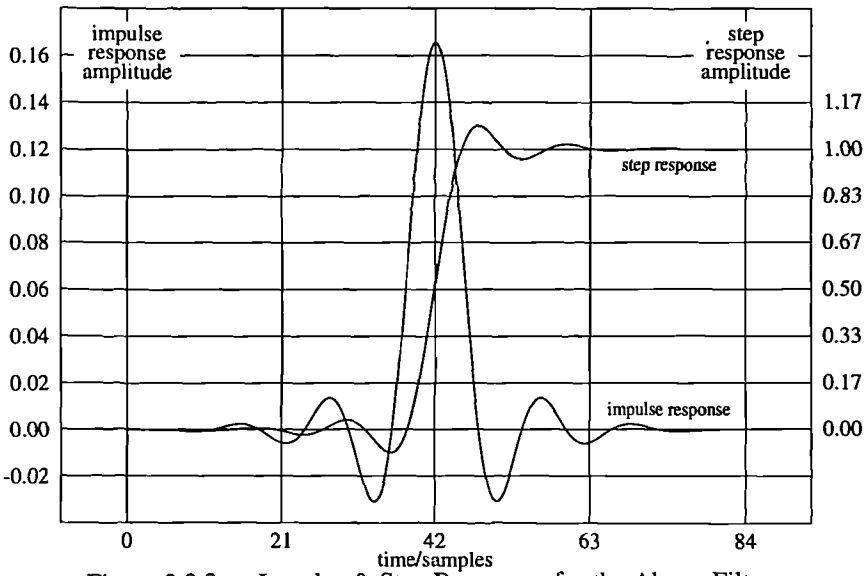


Figure 3.3.3.c : Impulse & Step Responses for the Above Filter.

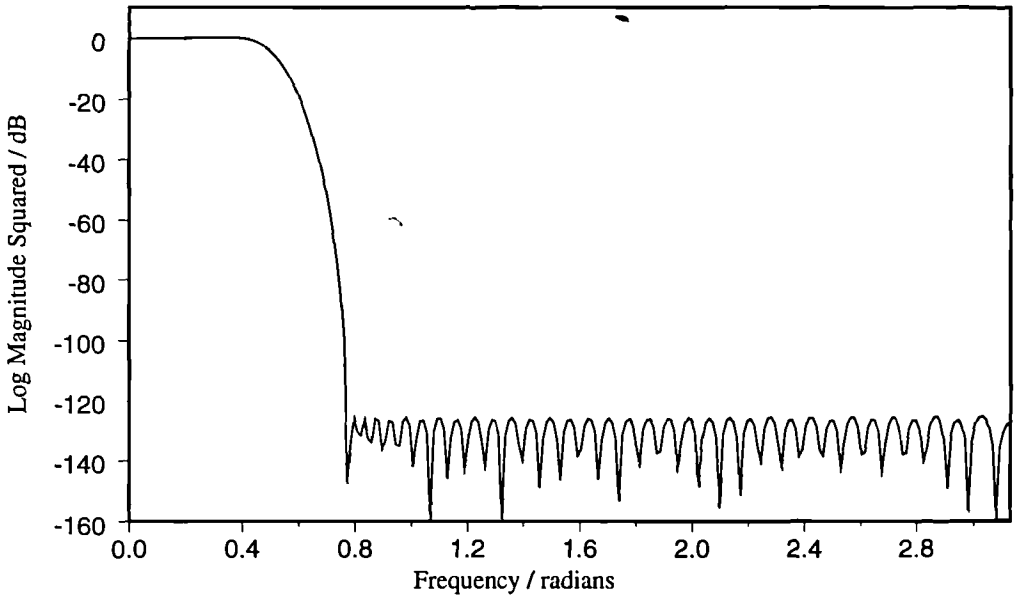


Figure 3.3.3.d : Frequency Response for the Above Filter.

3.3.4 : Estimation and Design of Classical Model, IIR Filters

In IIR filter design for decimation and interpolation, simple low pass filters can easily be used based on closed form equations for the classical types †. Once an analogue filter has been designed from the closed form description (eg. Butterworth, Chebychev I & II, Thomson or elliptic), transforms such as the bilinear transform can be used to convert the continuous transfer function of the analogue filter into the discrete transfer function of its equivalent digital filter. The transforming leads to warping of the frequency scale which has to be pre-compensated for (and allowed for in filter length estimation).

Design programmes for the classical analogue filters and their conversion to digital equivalents are commonly available (eg. 'Hypersignal') and the theory behind them is well known so these will not be discussed further (see [OPP75] pp.197-237, 268-269).

3.3.5 : Design of Low Pass, IIR Filters using Minimum-ρ Synthesis (with Arbitrary Magnitude and Group Delay Specifications)

For special applications such as in the feedback stages of closed loop PWM DACs, high computational speed and near constant group delay (in the passband) require special IIR filters designed for both magnitude and group delay specifications. A programme for this has been developed [DEC72] which outputs coefficients a,b,c,d and k, for a cascade of biquad sections as shown in figure 3.1.2.b.

The programme operates by applying a user defined error function which assesses the usefulness of a trial filter in terms of magnitude and delay requirements. The requirements are passed to the programme in two ways; firstly as a parameter file and secondly in the definition of the desired group delay and frequency response (written by the user). Successively better filters are produced using a steepest descent optimiser [FLE63] until no appreciable improvements can be made.

† IIR designs do not naturally exhibit linear phase and are not suitable if this is required.

In the process of finding the best combination of coefficients, the optimisation may make use of poles which are outside the unit circle, yielding unstable filters. To avoid considering these poles as useful, they are 'inverted', replacing the unstable pole with one of reciprocal length and similar angle (as plotted on the complex Z plane); this does not affect the magnitude response but has the direct effect of producing non-linear phase response.

This programme, 'deczky.for', has been used for most of the IIR filter designs used in this thesis and an example filter's pole-zero plot, group-delay and frequency responses, before and after optimisation are shown below. †

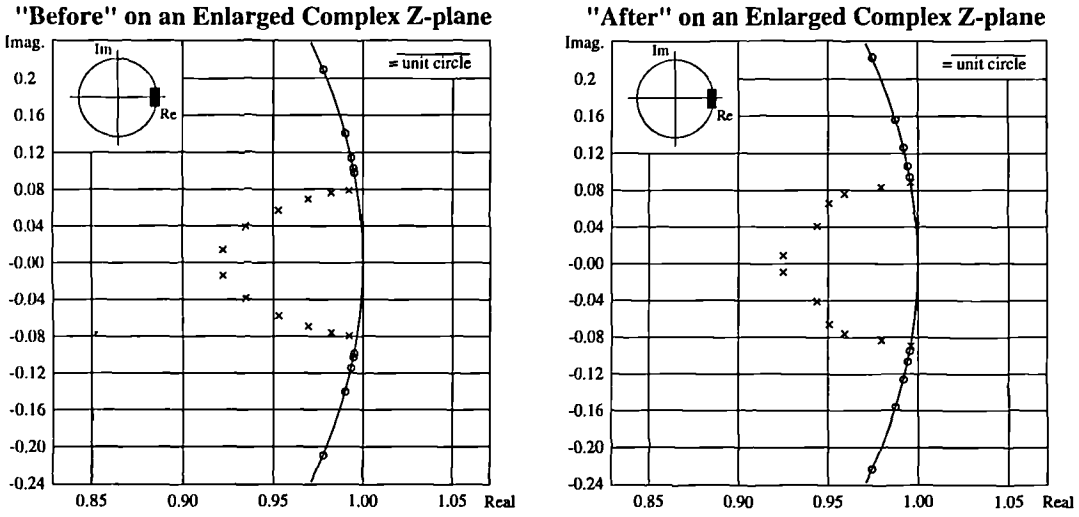


Figure 3.3.5.a : Pole-Zero Plot for a 12 th. Order Low Pass IIR Filter Before and After Optimisation.

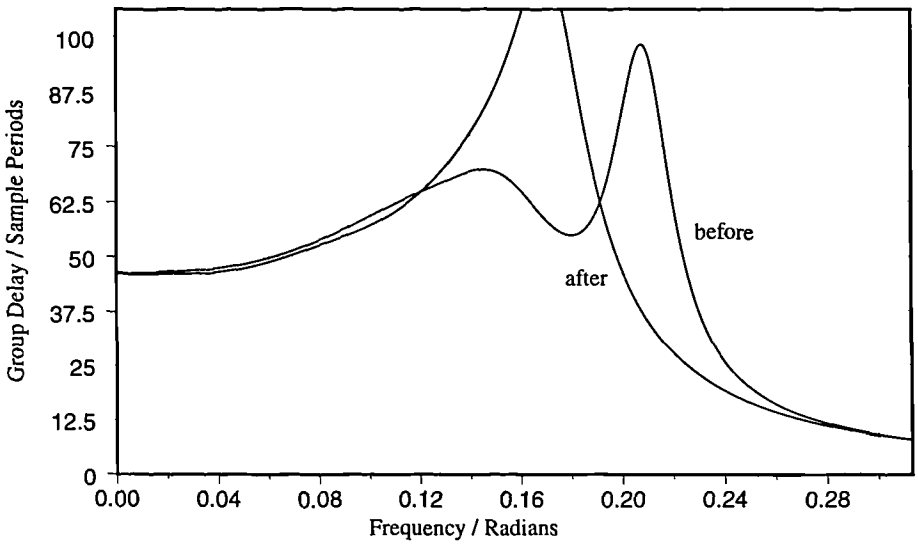


Figure 3.3.5.b : Group Delay Response (Before and After Optimisation).

† This example filter is used in section 5.4.2 ; the region in which the group delay needs to be controlled is from 0 - $\pi/144$ radians (0.02^c). Other regions are not constrained, hence the higher group delay which has been permitted near 0.17^c after optimisation.

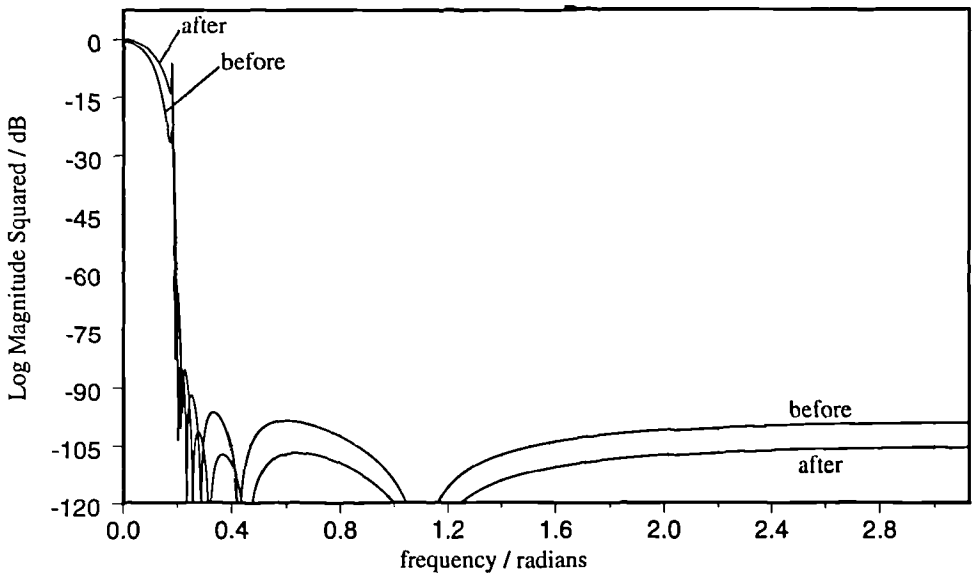


Figure 3.3.5.c : Frequency Response (Before and After Optimisation).

Optimised designs have been proposed [MAR79] to take advantage of the best of FIR and elliptic IIR designs, with a denominator as shown in equation 3.2.2.b, zeros on the unit circle, and equi-ripple passband and stopband behaviour. Computation rate reductions of between four and eight times have been achieved with this strategy (compared to the FIR approach) for a 20x single stage decimator. In a multi-stage implementation this is reduced to an improvement by a factor of only 1.2 .

Modifications can be made [RIC82] to the original optimisation routines [DEC72] to reflect these changes, but the resulting filters exhibit a compromise between the magnitude and group delay responses that forces higher order filters to be used, negating the benefits of the approach [MAR79]. This characteristic is typical of many multiple-criterion optimisation approaches where a set of filters (the 'non-inferior frontier') can be found with 'best combinations' of, say, group delay and frequency response. These difficulties will not be discussed further here (see [COR84], [NIC91]).

3.3.6 : Low Wordlength Coefficient Filter Design

When either FIR or IIR filters are implemented, distinct computational improvement can be made by operating with integer mathematics and preferably low wordlength mathematics. Design of filter impulse responses based on such mathematics is difficult and usually requires optimisation with the implementation constraints written into the optimisation preferences and is not efficient with high orders. These preferences are different for dedicated hardware implementation and software implementation (eg. on DSP integrated circuits or in simulation) since processes such as bit shifting in software require computation time whereas in dedicated hardware this can be implied by the connections used. The driving forces for the balance of preferences are cost and speed, neither of which varies smoothly, so each design using filters of this kind should be approached as a special case.

By altering the design philosophy or target filter structure better techniques can be found to design low wordlength coefficient sets. For example, if the filter structure can be modified to allow time variant coefficients, even IIR filters can be constructed which can be totally integeric and

multiplier free [GHA91]. One simple approach is to convert the high precision impulse response to a lower precision response by error spectral shaping (ESS, see chapter 4) and this has been found effective for high order filters [NIE89].

The idea behind ESS is that the effects of quantisation of the impulse response (ie. the FIR coefficients) can be placed in frequency bands that are not critical. In most cases the transition band shape is unconstrained, and the passband ripple is larger than the stopband ripple, thus if the process of quantising the filter coefficients places errors in these bands, small changes in the impulse response can be introduced allowing far easier implementation. The new FIR will have the same order but may lose half-band properties (if the floating point version was half band). The design method is outlined below.

(1) The impulse response is split at its centre so that after coefficient quantisation it can be reformed to ensure linear phase and hence enable efficient filter operation. If the original filter is not linear phase this step can be omitted.

(2) A noise transfer function (NTF) for the noise shaper is designed to suppress quantisation effects in the critical bands and replace them with minimum gain in the non-critical bands (this is described in more detail in chapter four). This can be derived from the filter itself, or designed specially, and can have full floating point precision since it is only used in the design process.

(3) The split impulse response is passed through a high order noise shaper, initialised with random errors with a rectangular probability density spread across the feedback range (± 1/2 LSB).

(4) The filter is reconstructed and its frequency response is checked to ensure it is still meets design requirements, repeating stages (3) and (4) until a satisfactory filter is found.

Using this technique, at least two bits of the rounded impulse response † can be reclaimed and more for some filters, especially those with high order or low dynamic range frequency response. Programme "nsfround.f" (see appendix A.3.6) applies this technique and was found particularly useful for noise shaper NTFs and for this application the NTF can be used to modify itself. An example of a floating point filter and its truncated and ESS processed frequency responses are shown below :

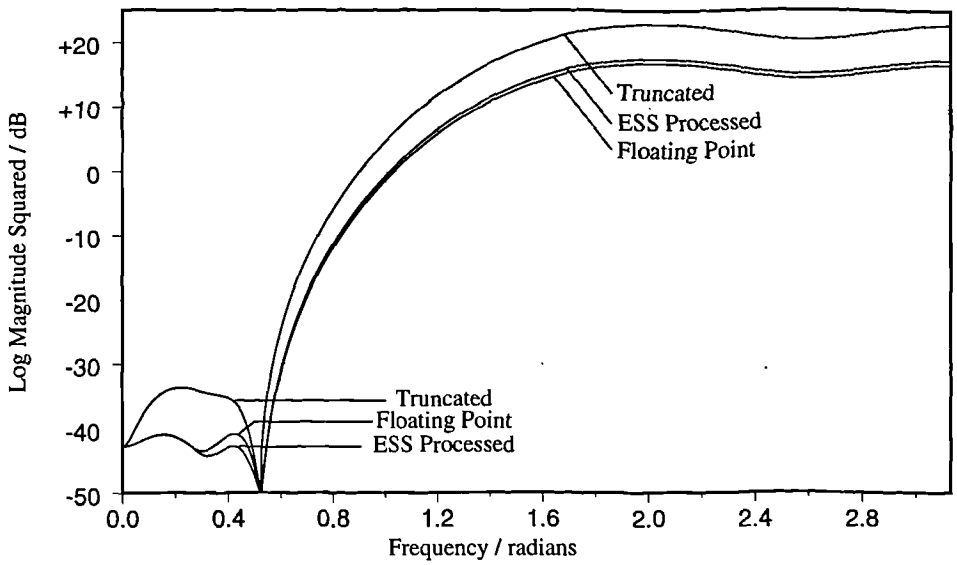


Figure 3.3.6.a : Frequency Responses of a Floating point, Truncated and ESS processed FIR Filter.

† The rounded impulse response can be found by scaling the coefficients until the smallest is normalised, and then rounding the coefficients to the nearest integer.

3.4 : Advanced Multi-Rate Filtering Techniques, A Case Study of Decimation for PWM Analysis.

3.4.1 : Overview

Once the benefits of basic multi-rate filtering have been taken advantage of (filtering at the low rate, using multi-stage filtering, exploiting filter symmetry, etc.) the improvements that can be made are largely dependent on removing redundancy due to special knowledge of the signal or filtering used. Good solutions have to be assessed on an individual basis, so in this section a case study of further improvements used in decimation for analysing the output of a DPWM is used to demonstrate some further improvements that can be made. This discussion is not exhaustive but serves only as an example of what benefits can be obtained with some imagination.

3.4.2 : Using PWM with Multi-Band (Comb) Filtering

When a PWM signal is to be filtered, as in decimation before spectral analysis, the nature of the signal can be used to simplify the initial stages of multi-stage decimation. Taking, for example, 288 level trailing edged modulation running at 8x oversampling with 88% modulation depth: the signal is known to to be high in only the first part of the sample period, and the state of the first 17 cycles in each period is known to be one and the last 17, zero, and the signal is known to only take values of one or zero. A diagram of this is shown below :

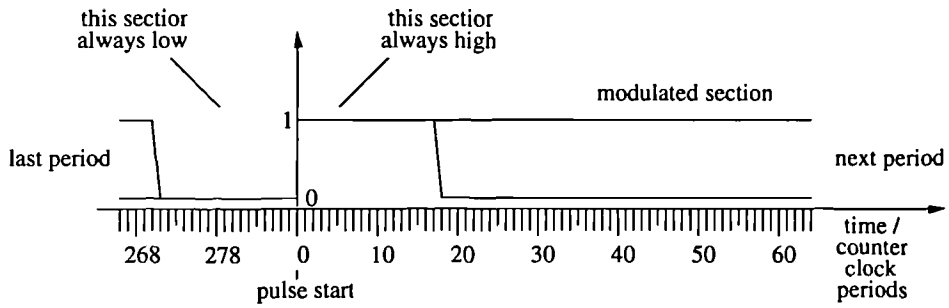


Figure 3.4.2.a : Known States of a Trailing Edged DPWM Output

A high overall ratio of sample rate change is required (8 x 288 = 2304) so comb filtering can be applied. Noting that 2304 has many factors of two in it ($2304 = 3^2.2^8$), suggests the use of half band filtering in subsequent stages; this leaves the other factors to define the initial decimation ratio and hence the length of the comb filter to be used; ie. the length should be a product of 3 or 3^2 and 2^n ('n' to be chosen). Since comb filtering is considerably easier to apply than more complicated filters, the largest ratio which does not lead to aliasing the audio band is should be used. First order combs only offer 60 to 80 dB suppression for significant first stage ratios (9 or 18) so a second order comb was chosen using two 18 tap combs. A diagram of its impulse response is shown below :

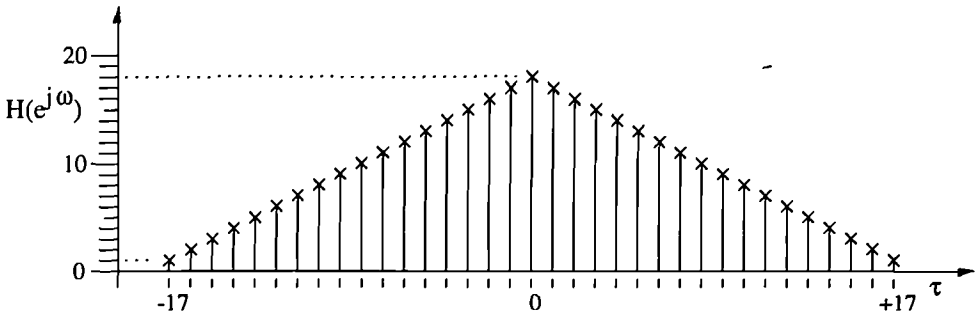


Figure 3.4.2.b : Impulse Response of a Second Order, 18 tap Comb Filter.

Since the combs' output sample rate is a factor of the combs' input data rate, a constant number of output samples will be produced per pulse. As the pulse is considered to be formed of 288 samples and a decimation ratio of 18 is being used, 16 output samples are produced per pulse. By aligning one of the output sampling points with the pulse start, the convolution of the decimating filter impulse response (as in figure 3.4.2.b) uses data from the known part of the pulse (see figure 3.4.2.a). The output sample for this is known in advance and need not be calculated. The remaining fifteen output samples will depend on the pulse width but their value can still be calculated in advance and stored in a look-up table. Thirty six filter outcomes can be addressed since there are thirty five taps in the second order comb, and there is the possibility that for a small pulse there will be no 'overlap' in the convolution of the filter with the pulse (ie. an output value of zero). By taking the quotient of the pulse width divided by 36, the number of output samples from complete 'overlap' can be found, the remainder of which can be used to define the next two output samples. The following samples are then set to zero up to the quota of sixteen for the period.

By passing data representing the number of overlaps, a significant reduction in the amount of storage space can be achieved both by reducing the numeric value passed (one byte instead of two), and by postponing explicit representation of the signal amplitude until after the next decimation stage (by which time the amount of data has been reduced). In this way, a simulation using $4^3/_4$ M.samples of data representing a 16-bit resolution signal can be stored within the capacity of base memory of a standard PC (640 kB) and still leave space for the operating system and programme code. This avoids slow disc access, and allows use of the faster memory which can be accessed without extended addressing.

The pulse and filter convolution stored in the look-up table can be easily evaluated since it has odd symmetry and the first half is an arithmetic series. Although the output pulse is conceived as either 'on' or 'off' (ie. one or zero), this leads to a DC offset which forces the use of larger wordlength processing in subsequent filtering. To avoid this, the offset is removed in the look-up table (representing the pulse as a bipolar NRZ type). A section of programme code for producing this convolution is given below and the contents that this would produce below that :

```
PROGRAM / PROCEDURE .... ;

VAR   comb  : ARRAY [0..35] OF integer ;
      index : byte ;

BEGIN
        ⋮
      FOR index := 0 TO 17 DO
      BEGIN
            comb[index] := (index * (index+1)) SHR 1 - 162 ;
            comb[35-index] := - comb[index] ;
      END ;
        ⋮
END .
```

Figure 3.4.2.c : Pascal Code for Calculating the Look-up Table Contents

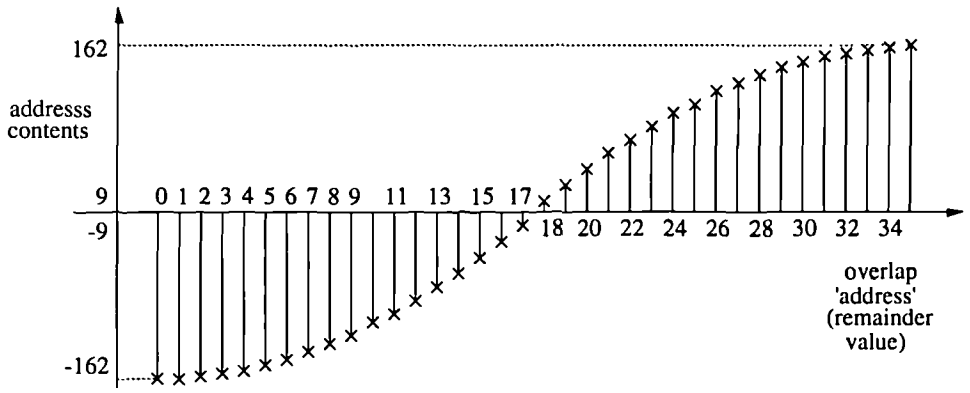


Figure 3.4.2.d : Look-up Table Contents, Plotted as a set of Modified Sample Amplitudes vs. Address

3.4.3 : Primitive Operator Filtering

In order to speed up the operation of filtering, integer mathematics should be used (ie. pre-scaled, fixed point maths requiring only integers) and special coefficients that can be implemented by addition and shifting should be employed [BUL88]. For simulations of closed loop PWM DACs (see chapter 5) and analysis of the output of open loop PWM DACs, a group of filters based on such techniques were developed with these properties.

After initial comb filtering of the PWM output, an integeric signal of 6 bit wordlength is produced, leaving a further 16x decimation to complete, to reverse the sample rate change produced by 8-bit DPWM (288x decimation overall from 254 data time-slots and 34 guard-band time-slots). The next stage of decimation handles a large amount of data and reduction of the complexity of operation of this filter greatly speeds up the overall operation. By taking an integer half-band filter from a well known family of half-band filters with special properties [GOO77], and convolving it with itself twice to generate a new filter, a filter with simply evaluated coefficients is constructed with three times the suppression (in dBs). The coefficients, modified filter structure, impulse and step responses and frequency response of this filter are shown below :

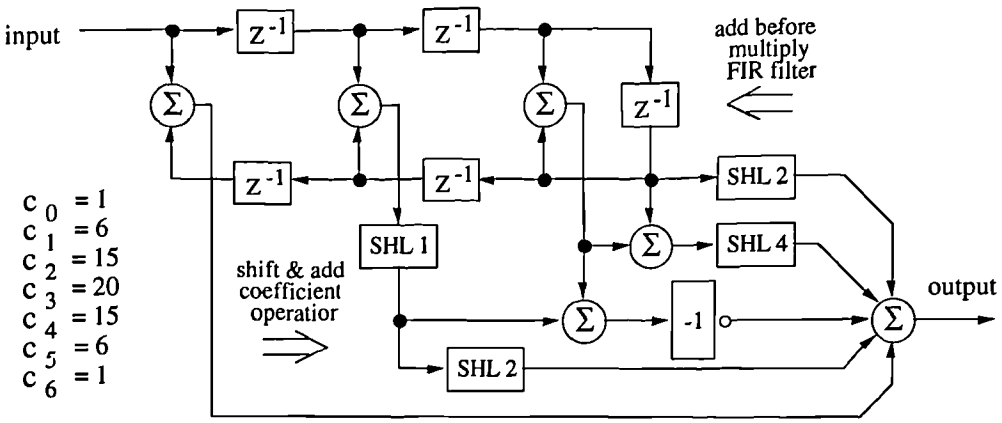


Figure 3.4.3.a : Coefficient Set and Modified Filter Structure for Fast Filter Evaluation

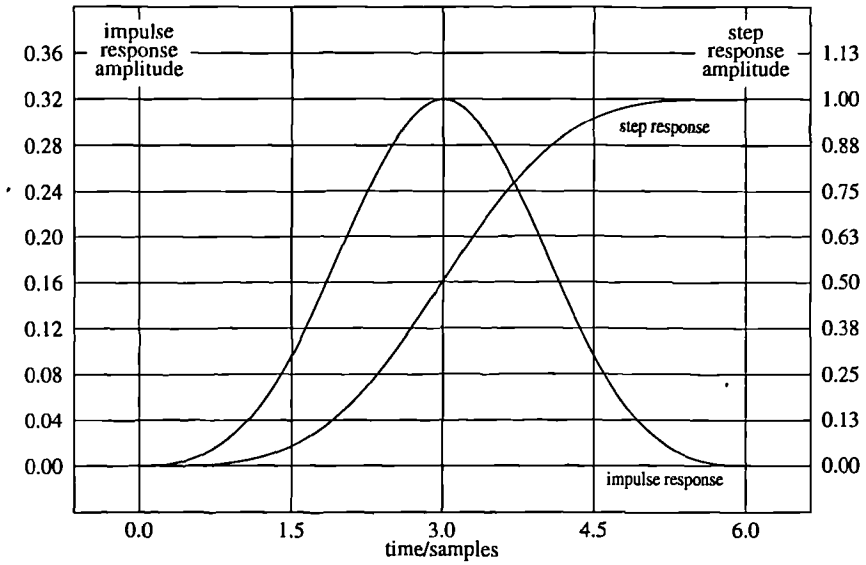


Figure 3.4.3.b : Filter Impulse & Step Responses.

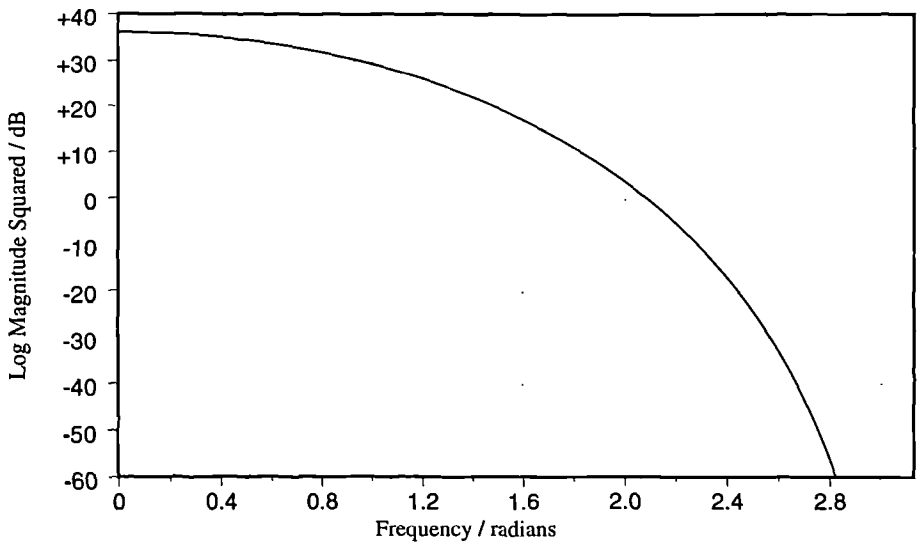


Figure 3.4.3.c : Filter Frequency Response.

3.4.4 : Multi-Stage Half-Band Filtering

If suitable pre-filtering can be applied in a multi-stage decimator or interpolator to allow the subsequent stages to be purely factors of two, half-band filtering is particularly suitable [BEL74]. Having completed a factor of 36x reduction from the 2304x overall by using comb filtering and half-band filtering implemented with primitive operators, 64x decimation remains to isolate the baseband, some of which can be achieved more quickly by long FFT than by sharply filtered decimation. An assessment of the ideal point at which to give up multi-rate filtering and apply an FFT involves a balance between the decimation processing delay time and the FFT processing delay time.

Doubling the length of the FFT of the output allows a factor of two reduction in the decimation ratio used overall for the same resolution in the spectral output over the band of interest. For the final stages of decimation this is a great saving of computation time since their transition bands are particularly narrow, but for earlier stages of decimation this is not the case since as the FFT processing increases (approximately proportional to: $n.Log_2|n|$) so the required decimation effort decreases (and more and more quickly because the filters become simpler and simpler). Using the internal processing of a spectrum analyser (Audio Precision's 'System One, Dual Domain'), a good balance was found empirically, using a 16384 point FFT (and a four term Blackman Harris window) after 8x decimation. This enabled the use of three half-band filters with 15 taps, 27 taps and 163 taps respectively. The frequency response of these is shown below :

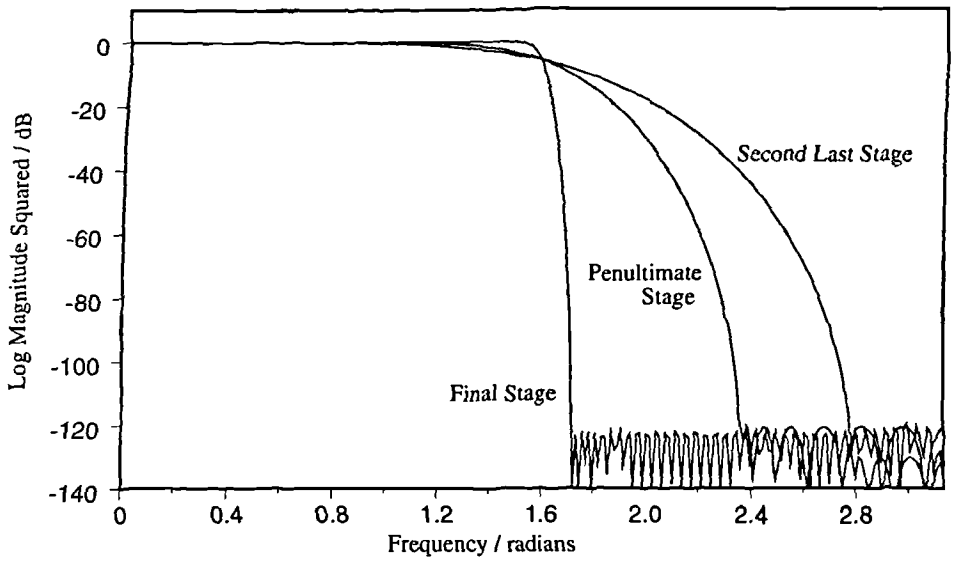


Figure 3.4.4.a : Frequency Responses of the Last Three Half-Band FIR Decimating Filters

More aggressive filtering than absolutely required was used to widen the un-aliased band and hence see more spectral features than was originally intended; for fastest operation this feature could be removed by replacing the last filtering stage by one with a wider transition band. These filters were implemented taking advantage of their impulse response symmetry, clocked at the output sample rate (the low rate - by only evaluating required output samples), and using arrays accessed in such a way as to allow overwriting of data ('in place data replacement'). This permitted maximal usage of the fastest memory available to the micro-processor (80286/87).

3.4.5 : Fast FIR Convolution

The direct form of the FIR filter implements the convolution of the filter and signal explicitly. This is not the fastest way of filtering since it exhibits redundancy, storing the signal many times. In some forms this enables improved filtering structures which take advantage of the explicit form and reduce the implementation redundancy but in the higher order FIRs these advantages are lost as floating point precision and arbitrary impulse responses are used.

Faster methods of convolution do exist which can be split into three broad groups: firstly, algorithms based on the transform-domain FFT approach, secondly, those applied in the time domain such as iterated FIR filtering [BLA87] (this operates by applying short Winograd convolutions and building a pyramid of these for longer convolution) or even a multi-rate decomposition employing multi-rate techniques [VET88]. Lastly, there exist algorithms based on exchanging multiplication cost for slightly increased addition cost by revising the convolution integral [CHE92].

FFT based algorithms do approach optimal performance but require additional control processing for bit reversal which makes them less efficient in software (although ideal for dedicated hardware). These routines also require large amounts of storage space making them unsuitable for the processing of large amounts of data in a PC environment. The iterated FIR filtering technique is slightly inferior to the FFT approach and still requires a large amount of control processing. The multi-rate technique makes an overall saving of 25%. When used in combination with half-band filters and techniques taking advantage of filter symmetry, significant savings (upto 480%) were achieved. Where filter symmetry was not available, moderate savings could be achieved using the revised convolution integral approach. This operates by rewriting the conventional convolution for input x(n), coefficients h(n) and output y(n) :

$$y(n) = \sum_{k=0}^{N-1} \{ x(n-k).h(k) \}$$

Equation 3.4.5.a

in three parts,

$$y(n) = \sum_{k=0}^{N/2-1} \{ [x(n-2k)+h(2k+1)].[x(n-2k-1)+h(2k)] \}$$
$$- \sum_{k=0}^{N/2-1} \{ h(2k).h(2k+1) \}$$
$$- \sum_{k=0}^{N/2-1} \{ x(n-2k).x(n-2k-1) \}$$

Equation 3.4.5.b

The last term can be evaluated efficiently with a recursive equation, the second term can be evaluated in advance, so the computational overhead can be reduced to 50% of the multiplies at the expense of ≈150% of the additions. Where multiplication is evaluated more slowly than addition (eg. in software simulation of decimation) this can represent significant savings in computation time.

3.4.6 : Intentional Aliasing for Examining Bands of Interest not including DC.

The severity of sideband performance from some DPWMs is extremely difficult to analyse since double Fourier analysis of the signal does not reduce to simple functions as is does in the uniformly and naturally sampled cases (see appendix A8). To assess the performance of such modulation types, simulation is a good route although numerical techniques can be used to overcome the shortfall in the theoretical analysis.

To examine sideband behaviour of PWM as well as the baseband, alternative filtering can be used to force aliasing of the high frequency band into the baseband after removing the original baseband content for analysis in the usual way. Having aliased the signal into the baseband, a conventional FFT can be used to zoom in on the area of interest, with the frequency axis modified to take into account the frequency modulation that has been performed.

The most significant contribution to baseband distortion from PWM induced sidebands is associated with the fundamental of the pulse repetition rate so frequencies around the PRR are of particular interest for characterising the distortion. When noise shaping is employed, this area is usually employed for dumping baseband quantisation noise so alternative noise transfer functions have to be used for this sort of analysis, with a noise stopband at high frequency (or band pass performance).

Taking an 8x oversampled signal as an example, PRR = 352800 Hz, and modulated by WAPWM, the primary sideband area is 352.8 ± 20 kHz. After appropriate pre-compensation and noise shaping, the signal can be modulated and comb filtered as before, but in each subsequent stage of decimation 372.8 kHz of baseband has to be protected. When the sample rate reaches 1411.2 kHz, the low pass half-band FIR has to be replaced with a band pass filter. As before, the transition bands are kept as wide as possible to simplify the filtering and each filter is operated at the low rate taking advantage of its symmetry. The spectra before and after the critical stage filtering and its following decimation up to the point at which the 16384 point FFT is applied are shown below :

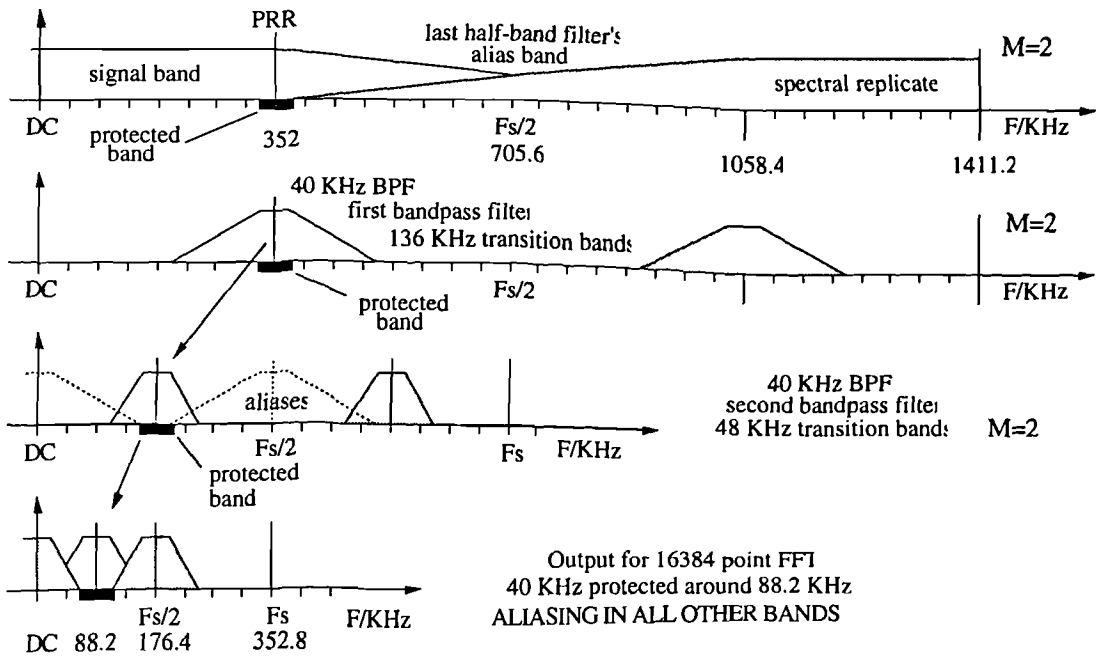


Figure 3.4.6.a : Intentional Aliasing Stage in Decimation for Sideband Analysis

3.5 : Summary

Sample rate conversion divides into two parts: interpolation (sample rate increase) and decimation (sample rate decrease). These are required whenever a digital signal is used in two parts of a system but at different sample rates, or to simplify narrowband filtering or spectral analysis.

Interpolation is achieved by the inserting of zero valued samples followed by digital filtering to suppress spectral replicates associated with sampling at the low rate; decimation (its dual) uses filtering first followed by the dropping of samples. For sample rate change by a factor of 'K', K-1 samples are inserted or deleted between the existing samples.

The required digital filtering can be achieved by recursive or non-recursive filtering with finite impulse response (FIR) and infinite impulse response (IIR), respectively. These filters require time to be implemented in either simulation or reality and so should be simplified as far as possible. Below, listed in order of significance are the main modifications that can be applied for optimal speed in FIR decimation and interpolation :

1) using multi-stage sample rate change,
2) operating the filter at the low rate (eg. only evaluating the output signals in decimation),
3) placing larger factors of the overall ratio first in decimation and last in interpolation,
4) partitioning multi-stage structures into the blocks of prime factors of the overall ratio,
5) taking advantage of impulse response symmetry by adding before multiplying out,
6) taking advantage of zeros in the coefficient set (eg. half-band filters in 'x2' stages),
7) applying multi-band filters (particularly comb filters to early filtering stages),
8) using look-up tables of precalculated filter output (especially special signals, eg. PWM),
9) using integer arithmetic, and special coefficients to allow shift and add implementation.

IIR filtering is more efficient than FIR in terms of operations per second, or data storage space requirements, but it has a non-linear phase response (as a function of frequency) and thus cannot be used in linear phase applications, and is more difficult to design as a result of additional stability constraints. IIR filtering can be made more efficient by items (1) to (4) and (9) above.

Standard, or optimisation design techniques can be used to design both FIR and IIR filter coefficients. In the IIR case, analogue filters of appropriate frequency response can be transformed to yield the required coefficients in the digital structure. In the FIR case, inverse Fourier transform of a suitable frequency response can be used since the coefficient set is a sampled version of the required impulse response. Optimal filters can be designed to satisfy various criteria. For spectral control, frequency domain based design is favoured, leading to filters with equi-ripple error in frequency.

Sample rate change is required for PWM DACs in five areas: interpolation, for increasing the DPWM linearity by increasing the signal to carrier frequency ratio, for enabling the use of simple output recovery filters and enabling the use of simple noise shaping (the subject of the next chapter). Decimation is used both for producing the feedback signals required in closed loop PWM DACs and for narrowband spectral analysis in the design stages of all PWM DACs.

Special filters with limited coefficient values (eg. 0, ± 1, $\pm 2^n$) offer greatly reduced computational complexity and some of these are exploited for a case study in decimation for PWM output analysis, along with the use of all of techniques (1) to (9) listed above.

Chapter 4 : Noise Shaper Design &~Performance

Overview.

In the first part of this chapter, the technique known as 'Noise Shaping' will be analysed and its requirements will be examined, so that appropriate filters and structures can be designed for a high resolution DAC. As discussed in section 1.4.1, noise shaping is a process whereby a limited output data wordlength can be used to express a high resolution input signal by concentrating quantisation noise in a separate frequency band from the signal. This process will be explained in more detail in the next two sections.

As shown in figure 1.1.1.a, noise shaping is used just before digital pulse width modulation in the proposed system for a digital amplifier. This allows the oversampled signal data from the previous stages to be expressed in a smaller number of bits which the DPWM can handle.

Noise shaping is used to make implementation of the digital amplifier easier. By reducing the data wordlength used in the DPWM, the counters within the DPWM can be clocked at a much lower frequency. Without noise shaping, clock rates of the order of gigahertz are expected for signals with audio bandwidth and resolution whereas with noise shaping, prototype circuits running at ≈100 MHz have been constructed [HIO91a].

For noise shaping to operate properly, a spare frequency band is required to put quantisation noise into. For audio reproduction this can be at high frequency because the noise will then be inaudible and may be filtered off by passive low pass filtering if necessary. This spare high frequency band is available in oversampled signals and can be provided digitally by interpolation (see chapter 3). Interpolation is used in many types of D/A converter to simplify output recovery filtering, so this requirement for the noise shaper does not make the system unusually complicated.

Noise shapers use a recursive filter structure to shape quantisation noise so that it lies in the spare frequency band. Simple high pass filters based on differencers can be designed which cause the output quantisation noise power to be large at high frequency. Although easy to implement, these filters have limited use because the high noise power at high frequency can reappear as a sideband in the audio band as a result of of the pulse width modulation process (see chapter 2). These filters will be looked at in detail in the second part of this chapter.

A balance can be struck between the high frequency noise power and the audio band noise power so that high resolution digital amplifiers can be constructed. This balance requires a more complicated filter characteristic, is less easy to implement and requires a sophisticated design procedure. In the third part of this chapter, the properties and performance of these filters will be demonstrated and a design technique will be presented.

Special filter design techniques based on various optimisation procedures and with various objectives will be discussed in section 4.4. Techniques for accelerating the optimisation will also be presented since this approach has been found to be computationally intensive. Implementation of the filters found by optimisation has been observed to reduce some of the spectral performance achieved in the design stages through rounding of coefficients. Additional routines and structures are also presented to avoid these problems.

In the penultimate section of this chapter, the design of special filter shapes for particular applications is explained. These involve mixed domain optimisation procedures, controlling impulse, frequency and group delay response characteristics simultaneously. These can offer even lower noise output or the opportunity for considerable processing in the feedback of a noise shaper as required in closed-loop PWM DACs; they will be referred back to in chapter 5.

The last section of this chapter deals with compound noise shapers constructed by series or parallel organisation of two basic noise shapers. So called 'Multi-quantiser' noise shapers will be shown to offer noise performance advantages in return for additional complexity and will be discussed again in chapter 6 where a prototype digital amplifier is described using one of these structures.

4.1 : Description of the Error Feedback Technique.

4.1.1 : A Time Domain Approach

As discussed in section 1.4.1, high wordlengths applied to PWMs imply internal counting rates many times that of the sampling rate. This arises because the edge of the pulse must be defined with a resolution fine enough to represent each input level with a different output pulse width. One simple way of achieving this is to truncate (or round) the high wordlength signal before applying it to the PWM, but this is equivalent to adding broadband noise to the signal and is unacceptable in a low noise system.

The noise introduced by truncating (or rounding) can be reduced by retaining a record of the error and adding it to the next sample of the input stream so that it is not 'forgotten'. This is error feedback in its simplest form, and may also be referred to as error spectral shaping (ESS) or noise shaping, a block diagram of which is shown below.

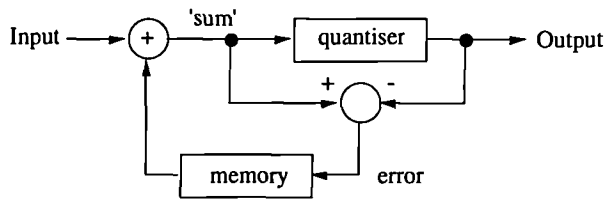


Figure 4.1.1.a : Basic Noise Shaper Block Diagram

Digital recursive systems such as this cannot react fast enough to accommodate all frequencies in the input stream since the error cannot be added to the input until the next sample, however at low frequencies the input can be 'tracked' very closely by the low wordlength output. To increase the usable bandwidth of the output, three options are available:

1) the output wordlength can be kept as long as possible,

2) the sampling frequency can be increased, and

3) the process can be done repeatedly.

Maintaining the output wordlength as long as possible may seem to defeat the objective of reducing the wordlength but it should be remembered that *any* reduction in the capacity of the channel being used to convey the digital signal will limit the information that the signal can convey, and minimising this is the primary objective of the overall D/A converter.

Similarly, increasing the sampling frequency of the digital signal may seem to be contrary to reducing the counting rate inside the PWM but as will be seen shortly, doubling the sample rate can enable wordlength reduction in the noise shaper by *more* than one bit, hence reducing the counter rate overall. This can be viewed as a result of not only having increased the channel capacity, but also reducing the feedback delay so that less deviation from the input (error) has occurred by the time it is accommodated for.

Repeating the noise shaper is not effective unless the wordlength is reduced successively since requantising a digital signal to its own wordlength will have no effect on it. Unfortunately, even repeated noise shaping using fewer and fewer bits at each stage is not very useful, since the noise from each stage will be increased by subsequent stages. This points to one useful conclusion, namely that using more errors than just the last, is useful for tracking the signal closely. By using several previous errors a higher order feedback loop can be developed and this will be analysed in the next section from a frequency domain point of view.

4.1.2 : Frequency Domain Analysis of a Noise Shaper.

The spectral characteristics of a noise shaper can be neatly summarised if the signal transfer function (STF) and the noise transfer function (NTF) are treated separately. To do this the output has to be assumed to be the sum of the input signal and an error term. Having made this assumption, simple analysis can be used to derive the NTF and the STF using a more general bock diagram of a noise shaper as shown in figure 4.1.2.a . The 'memory' has been replaced by an arbitrary function, H(z), which produces a linear combination of the previous errors, and the quantiser has been approximated by an additive error [TEW78]. This configuration and the variables from the diagram will be used in the analysis.

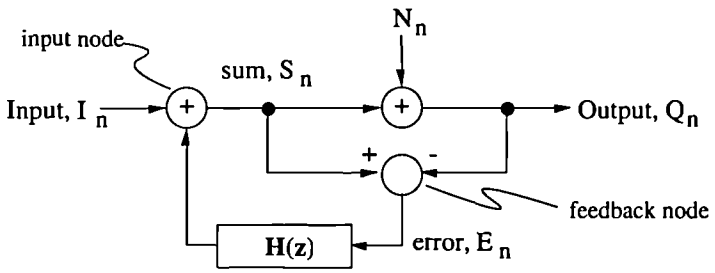


Figure 4.1.2.a : Equivalent Block Diagram For a Noise Shaper.

Evaluating at the input node,	$S = I + H(z) . E$	Equation 4.1.2.a
Similarly, from the feedback node,	$E = S - Q = -N$	Equation 4.1.2.b
and from the output node,	$Q = S + N$	Equation 4.1.2.c
hence, putting 4.1.2.a & b in c,	$Q = I + H(z) . E - E$	
or	$Q = I + (1 - H(z)) . N$	Equation 4.1.2.d

from this the error added (N_n) can be seen to have been modified by "(1 - H(z))" but the input has not been affected, hence : STF = 1, NTF = 1 - H(z) Equation 4.1.2.e

In the simple case considered in section 4.1.1, H(z) was simply a unity gain delay, so the NTF was that of a digital differencer (an approximate differentiator), which has a high pass characteristic. Arbitrary filters can be used in this feedback loop and will be examined in sections 4.2 and 4.3. Higher order differencers are particularly useful since as the order increases these yield wider usable signal band by shaping the requantisation noise to be increasingly at higher frequencies. Higher order differencers will be the subject of particular scrutiny in section 4.2 because these yield simple coefficients in the feedback filter H(z).

The general form of the NTFs described so far have been high pass filters because these are well suited to audio reproduction. Shown below is the general nomenclature for the various regions of the filter's frequency response which will be used throughout this chapter.

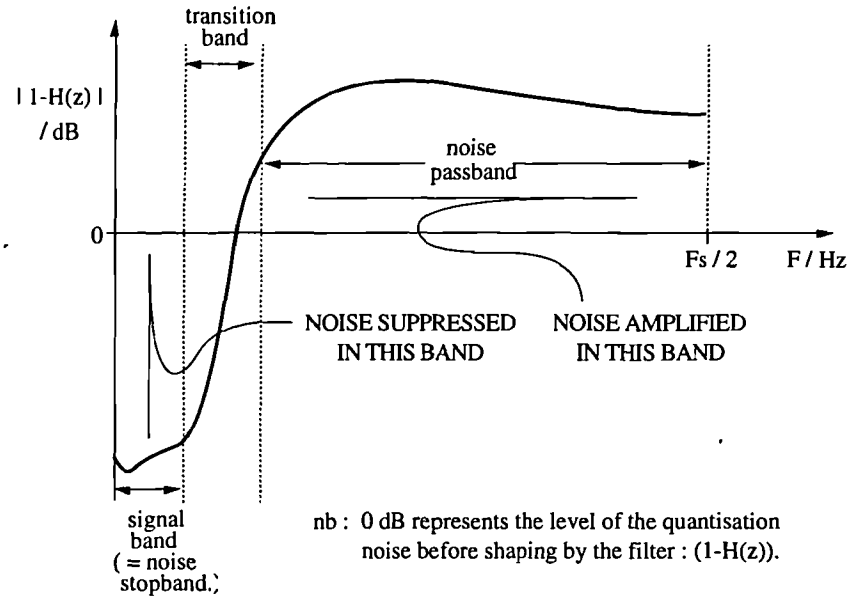


Figure 4.1.2.b : Noise Shaping Filter Frequency Response ("1-H(z)").

The validity of the analysis used so far hinges on the assumption that the effect of quantisation is one of adding an error. This assumption holds true for large input signals where the lower bits are not correlated to the higher bits but it does begin to fail in some interesting ways when small signals are applied to the noise shaper. More problems arise in this system if care is not taken to avoid output overflow because of increasing the input wordlength by adding the feedback signal to it. Small signal, and overflow considerations will be discussed in the next section.

4.1.3 : Overload Limitations in Single and Multi-bit Designs.

Simple though the noise shaper is, its operation can be spoilt because the quantiser is not as linear as the assumptions above made out. The severity of the quantiser's non-linearity is linked to the number of bits allowed in the output; when this is reduced, stability of the overall loop cannot be guaranteed by following the rules of linear analysis.

Presuming that overload should not be allowed to occur, a limit on the input signal size is

imposed because the feedback network may add to the input to send it into overload. For example, if we take a first order system as used in section 4.1.1, truncating 16 input bits to 8 output bits, the error signal can be as large as +255. In an input ranging from -32768 to +32767, this excludes input values above 32512, and assuming a symmetric input, reduces the maximum signal size to approximately 99%. While this may not seem severe, the example is one of only mild wordlength reduction and this problem becomes more acute with lower output wordlength. Furthermore, the simple first order differencer does not produce a very wide signal band and more useful NTFs tend to have higher gain in the feedback filter - this is a point of concern which will be investigated later.

To demonstrate how quickly overload becomes a problem, if a fourth order differencer is used (with a peak gain of 15 in H(z)), and a six bit output, the error signal can be as large as 1023, which after filtering can cause peak filtered error of 15360, leaving only 53% of the original input range available. Peak filtered error only occurs rarely, so many commercial designs avoid this issue by sensing overload and resetting the loop filter before the output becomes noticeably unstable. This introduces distortion which listeners describe as 'harshness' [CRA92].

When small output data wordlength is required, the quantiser non-linearity is so severe that loop orders greater than 2 usually prove unstable. This occurs on top of large overflow problems, and a large volume of work has been published on techniques to try to maintain stability under these conditions. While some concentrate on modified structures to avoid the stability problems [RIB91,CHA90], others optimise the loop filter to maximise signal-to-noise ratio in the audio band [HOR91]. One notable case [MAT89] even recognises the overflow condition and allows the 'carry' bit to be used with the signal to control a pulse width modulator. For the system proposed in section 1.4, multi-bit noise shaping will be used because low pulse repetition rate is one of the design objectives, so to minimise the added noise of the system the highest DPWM clock frequency should be used to permit using a large output data wordlength from the noise shaper.

4.1.4 : Idle Channel and Noise Modulation Effects.

As mentioned at the end of section 4.1.2, linear analysis of the noise shaper assumes the requantisation noise to be uncorrelated with the input. For small signals this is not a valid assumption. In practical systems noise correlated to the signal manifests itself in two particular ways,

1) the response to an idle (quiet) input channel is not necessarily an idle output channel,

2) the noise in the output will become a function of the input signal level.

Shown below is the output spectrum of a simple noise shaper using a first order differencer as its NTF. The input wordlength is 16 bits, the output wordlength is 8 bits and the input is set to the DC level of 000D (hexadecimal). Note how the lower frequency region of the noise shaper output spectrum shown below has become contaminated with harmonically related tones.

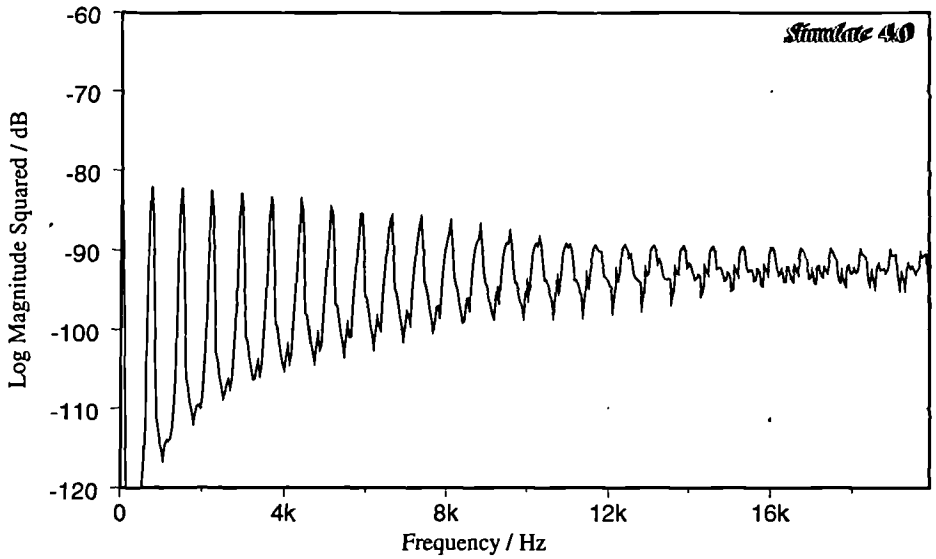


Figure 4.1.4.a : Idle Channel Tones From a Noise Shaper

Under idle channel conditions the human ear adjusts to a higher sensitivity, and there is no intended signal to provide spectral masking (see section 1.5) making spurious tones coming from the noise shaper easily audible. This is exacerbated by the spuriae being tones, and usually harmonically related ones, two features which the ear is particularly good at detecting.

The ear is more sensitive to noise modulated by a recognisable signal, than it is to steady noise of a higher power. Consequently, it becomes worthwhile to *add* a small amount of noise to the output of the noise shaper in order to avoid the noise modulation and idle channel tones. The signal below is the result of running a small signal added to a slow ramp through a noise shaper as used above. Note the 'beating' of the noise envelope which is introduced by the noise shaper.

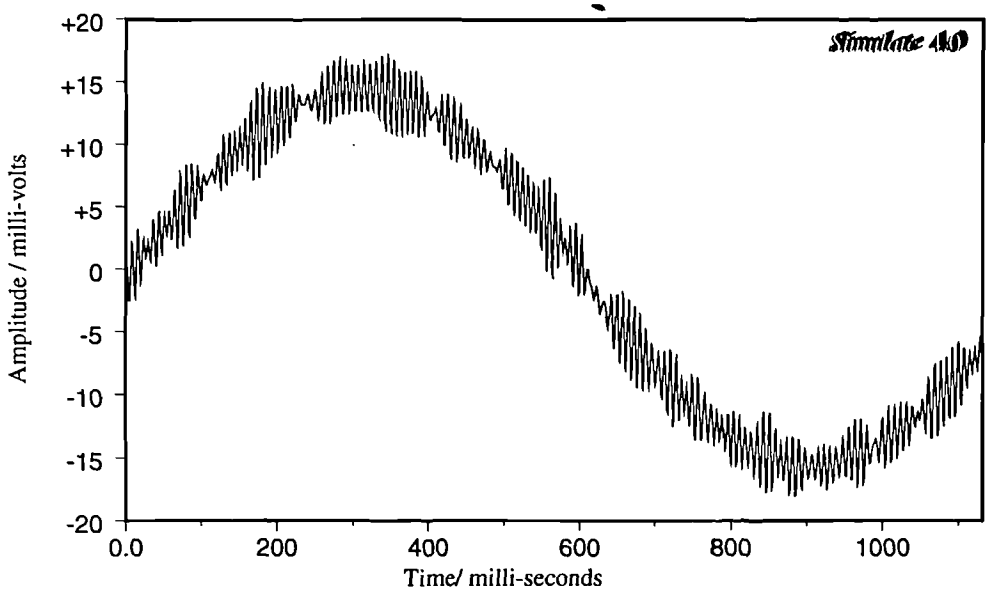


Figure 4.1.4.b : Noise Modulation From a Noise Shaper

Adding a small amount of noise just before the quantiser can be shown to decorrelate the error signal from the input signal [VAN89]. This avoids the small signal errors that the ear is sensitive to in favour of larger noise which the ear is less sensitive to. Such a signal is called 'dither', and is desired at any point in a DSP chain where quantisation occurs. In a noise shaper, this signal can either be added to the input with the feedback signal or just before the quantiser and in the next section these two insertion points will be shown to be equivalent under certain conditions.

4.1.5 : Comparing Dither Insertion Points of Simple Noise Shapers.

Dither is required wherever quantisation or requantisation is used in a signal processing chain. In the case of a noise shaper which employs requantisation (taking an input quantised to one wordlength, and quantising it again to an output of lower wordlength) the quantiser should be correctly dithered. It would be natural to assume that the dither should be added just prior to the quantiser since this is the signal that needs to be dithered, but it may be more convenient to add noise to the signal, and thus inject surplus noise into the feedback loop. The two alternatives are shown below:

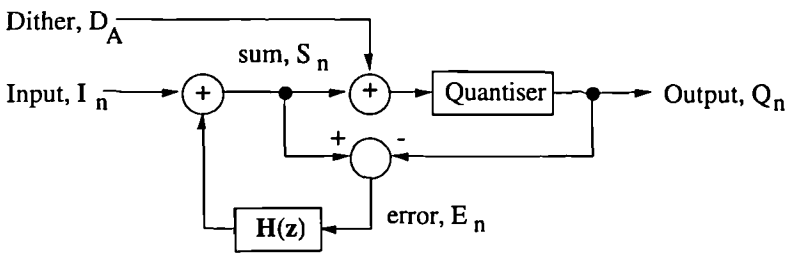


Figure 4.1.5.a : Dither Inserted Just Before the Quantiser.

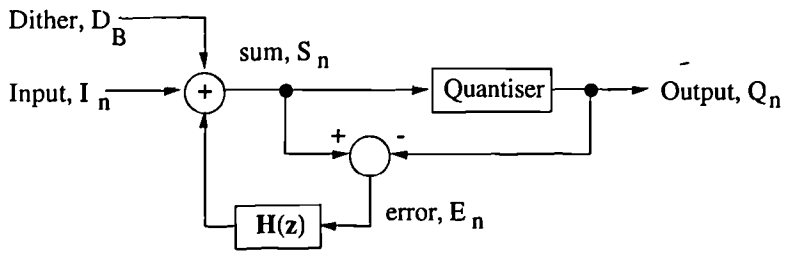


Figure 4.1.5.b : Dither Added to the Input Stream.

These two configurations can be shown to be the same, under normal signal conditions, by moving the nodes to form an equivalent network. Taking figure 4.1.5.a, the dither inserted just before the quantiser could be imitated by the same signal being added to 'sum' instead, provided it is also subtracted from the error, leaving only the feedforward signal around the quantiser changed (and this will not be used in the analysis). This equivalent network is shown below:

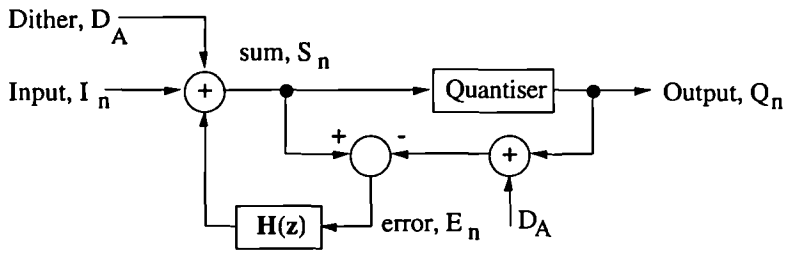


Figure 4.1.5.c ; Equivalent Network to Dithered Noise Shaper Shown in Figure 4.1.5.a

From this we can easily see that if the relationship between dithers 'A' and 'B' is :

$$D_B = D_A - H(z).D_A \qquad \text{Equation 4.1.5.a}$$

This is particularly interesting because it implies that equivalent dither can be applied outside the noise shaper provided it has been filtered by the NTF, 1 - H(z). Conversely, it could be said that dither added just before the quantiser will have the same effect on the output as adding a pre-filtered noise signal to the input. Also, the dither transfer function can be seen to be the same as the NTF and therefore that wideband noise added just before the quantiser will not be as damaging as adding wideband noise to the input stream, since it will be shaped by the NTF and appear predominately at high frequencies in the output.

Shown below are the output spectra from the same noise shaper used to produce figure 4.1.4.a with wideband dither inserted just before the quantiser, (figure 4.1.5.d), and at the input (figure 4.1.5.e).

Note, neither exhibits idle channel tones (and neither would produce noise modulation although this does not appear in quick spectral measurements) and the lower noise floor is in the first case, which is the best additive dither point if no pre-filtering of the dither is to be used.

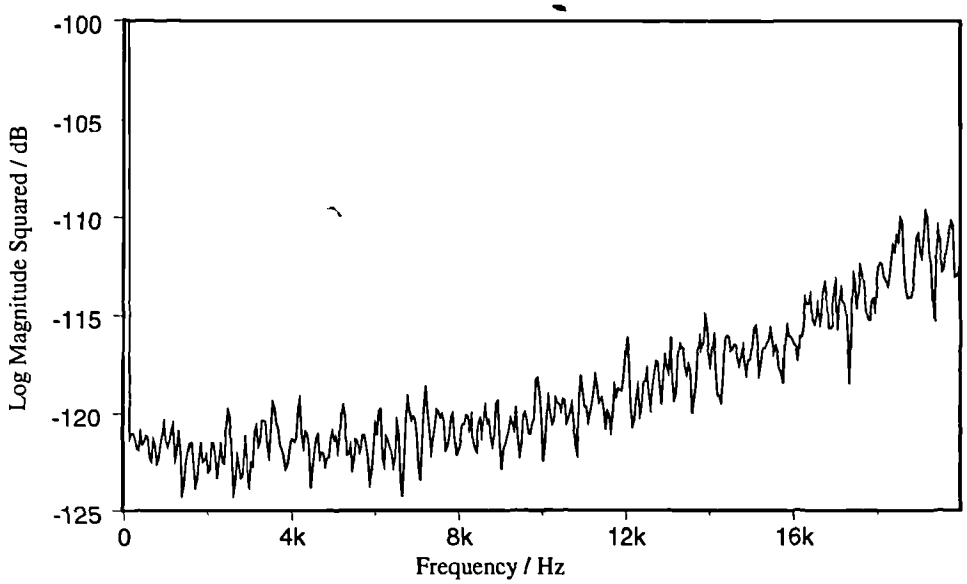


Figure 4.1.5.d : Additively Dithered Noise Shaper Using 'Node A'

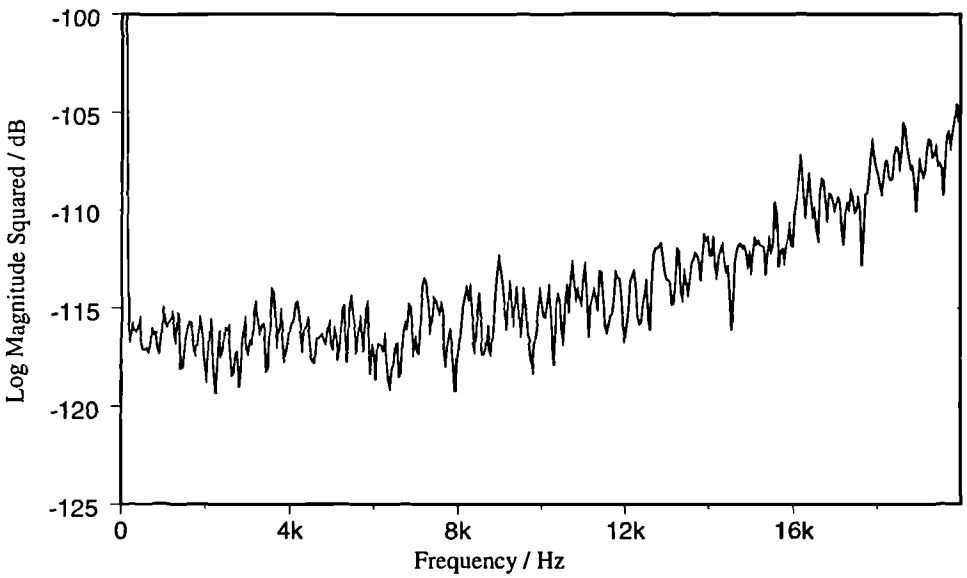


Figure 4.1.5.e : Additively Dithered Noise Shaper Using 'Node B'

4.1.6 : Noise Shaper & ΣΔ Modulator Similarities.

Alternative configurations exist for error spectral shaping, namely 'sigma delta modulators' (ΣΔ modulators). Functionally these operate in the same way as a noise shaper, but the topology is modified to enable easier implementation, especially for A/D converters. While these may use the same NTF, H(z) is modified to G(z) and placed in the forward path of the structure as shown in figure 4.1.6.a. To maintain the same NTF this implies G(z) must be an integrator, and because it is now seen by the signal, the STF is no longer unity. Typically, G(z) is a low pass filter (contrast with H(z) which was a high pass filter) and thus in the passband its gain is near unity, so for low frequencies, STF ≈ 1. Equivalence between these two networks will be examined in section 5.4.3 .

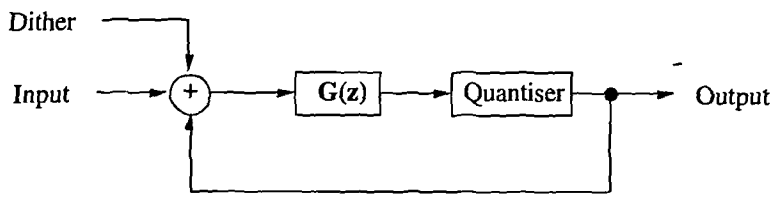


Figure 4.1.6.a : ΣΔ Modulator Structure For Error Spectral Shaping.

This network is particularly useful for A/D conversion if G(z) is replaced by G(s), a single bit quantiser is used (comparator) and a one bit D/A is placed in the feedback path, eliminating many of the problems associated with multi-bit ADCs. It is also interesting to note that the input will usually self dither with inherent noise as it's an analogue channel, avoiding the need for a dither source.

4.2 : Design and Performance of 'Sinusoidal' NTFs.

4.2.1 : Designing NTFs for Minimum Audio Band Requantisation Noise.

In section 4.1 it was shown that NTFs that were based on digital differencers were particularly useful for noise shaping. In high resolution systems, retaining as much of the input SNR as possible is one of the main factors in the choice of suitable DSP. In the noise shaper, this implies finding an NTF which introduces as little requantisation noise as possible into the signal band without limiting the input range too severely. Digital differencers have been proposed for this task [TEW78] since they are easy to implement, use integer coefficients, and introduce a falling level of requantisation noise at frequencies approaching zero (ie. very good noise suppression for small signal bands). This is of particular interest when ESS is used in conjunction with A/D or D/A conversion since increasing oversampling at the input can increase the dynamic range to the practical limits of the digital channel, and low pass filtering of the analogue output can remove the high frequency content arising from any noise shaping used. This enables the mismatch of components within multi-bit A/D and D/A converters to be avoided by using single bit designs, and in many cases anti-alias filtering and sample and hold circuits can be eliminated [CAR87]. If the high sample rate is inconvenient, multi-rate digital filtering can be used to reduce the sampling rate by decimation.

High order differencers have repeated zeros at DC since their NTF can be simply expressed as :

$$NTF = (1 - z^{-1})^{n} \qquad \text{Equation 4.2.1.a}$$

Provided the requantisation noise spectrum is assumed to be white [CAN85,OPP75] the shaped requantisation noise in the output spectrum can be approximated [RIB91] by:

$$S_q(\omega T) = R_q . [2 \sin (\omega T/2)]^{2n} \qquad \text{Equation 4.2.1.b}$$

where R_q is the unshaped requantisation power spectral density (PSD) and 'n' is the order of the differencer. This type of noise shaper has become known as 'Sinusoidal' as a result of this shape of PSD (see figure 4.2.1.a below), and they are particularly common in oversampled noise shapers (and $\Sigma\Delta$ modulators for A/D conversion; these will be discussed shortly).

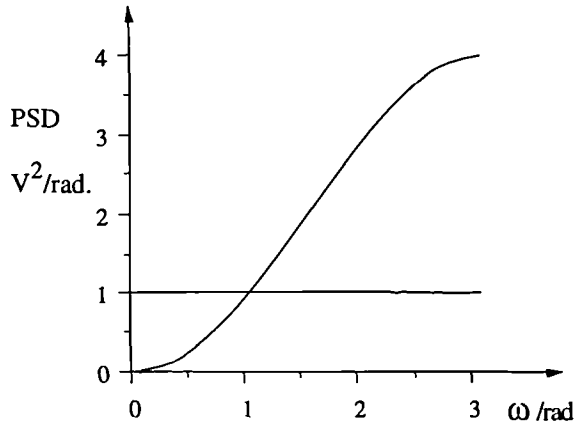


Figure 4.2.1.a : PSD of Requantisation Noise Before & After Sinusoidal Noise Shaping.

By integrating $S(\omega T)$ over the frequency band of interest, the requantisation noise contribution after decimation can be calculated and from this the potential output SNR can be found [CAN86].

Making the 'white noise' assumption, noise with standard deviation, σ_q, is spread evenly with PSD:

$$R_q = \sigma_q^2 . T/\pi \quad V^2/Rad. \qquad \text{Equation 4.2.1.c}$$

after shaping by an NTF of the form 4.2.1.a,

$$S_q = \sigma_q^2 . T/\pi . [\, 2 \sin(\omega T/2) \,]^{2n} \quad V^2/Rad. \qquad \text{Equation 4.2.1.d}$$

integrating with respect to frequency from DC to some portion of the band, 1/L,

$$\text{NOISE} = \sigma_q^2 \frac{T}{\pi} \int_0^{\pi/LT} (2 \sin(\omega T/2))^{2n} d\omega \qquad \text{Equation 4.2.1.e}$$

provided π/L is small, (ie. evaluating at low frequencies only),

$$\text{NOISE} \approx \frac{\pi^{2n}}{2n+1} \left[\frac{1}{L}\right]^{2n+1} \sigma_q^2 \; V^2 \qquad \text{Equation 4.2.1.f}$$

From this it should be noted that if either the order 'n' or the oversampling ratio 'L', is increased, the noise left in the band of interest (low frequencies) reduces until only that inherent in the input remains significant. It is worth noting that in the case of an oversampled A/D converter, the inherent noise power will be spread over a larger band with increased oversampling ratio. This leads to a reduction in the PSD by a factor of 1/L, which in amplitude terms increases the number of bits of resolution by half a bit for each factor of two increase in L.

In the case of an interpolated signal the baseband noise is not spread when sample rate increase is achieved by spectral replicate suppression, and the baseband noise PSD remains unaltered since it lies in the passband of the interpolating filters (which have a gain ≈1). This PSD, inherent to the input, sets the maximum SNR that can be achieved from the output of the noise shaper. In the following section the performance of various sinusoidal noise shapers will be demonstrated.

4.2.2 : Baseband Performance of Sinusoidal NTFs.

Using equation 4.2.1.d, it becomes easy to estimate the required order of sinusoidal noise shaper required to maintain SNR over a given band for a given wordlength reduction, but some loss of SNR must be tolerated. Defining what amount of SNR loss in the noise shaper is acceptable depends on the application and the available oversampling. A good compromise between system complexity and system quality can be struck, by forcing the requantisation noise at the high frequency limit of the band of interest to be less than or equal to the level as the inherent noise (assuming that is white). Using this criterion, and knowing the number of bits to be dropped (input wordlength - output wordlength) the oversampling ratios for each order can be plotted (see figure 4.2.2.a).

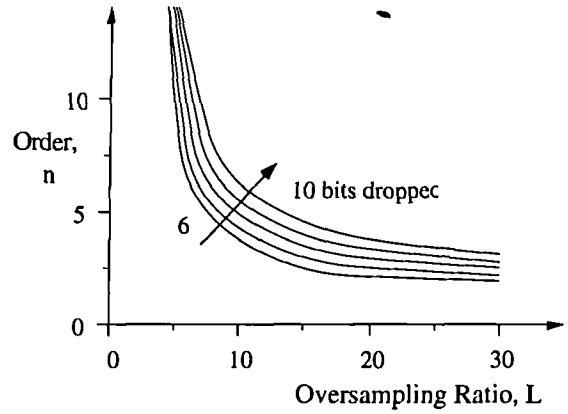


Figure 4.2.2.a : Equi-performance Contours of 'n' vs. 'L' for a Sinusoidal Noise Shaper (dropping 6,7,8,9 or 10 bits).

When the number of output bits and oversampling ratio are known, the internal clock speed of a PWM to follow the noise shaper can be calculated. The limit of this can be superimposed on a graph of order vs. oversampling ratio, thus setting a minimum requirement on the order. This has been done below for the case of 'bits dropped = 8' (see figure 4.2.2.b) but does not take into account what the influence of high frequency noise being passed to the PWM may do to the output, nor how close the system comes to overload; these problems will be addressed in section 4.2.4 .

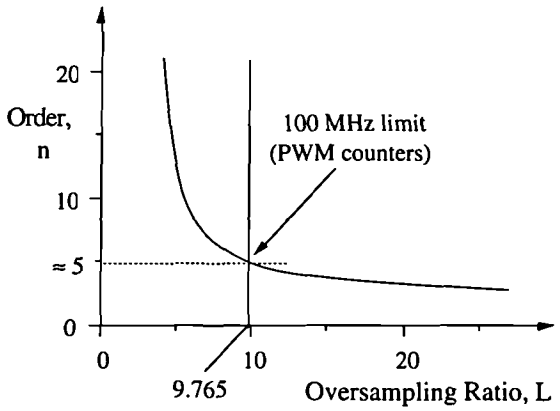


Figure 4.2.2.b : 'n' vs. 'L' for Sinusoidal Noise Shaping Dropping 8 bits at the PWM Counter Limit.

4.2.3 : 'Optimal' Dithering For Use With Higher Order Shapers.

Knowing that the dither signal added just before the quantiser of the noise shaper (node A) will be shaped by the NTF (see figure 4.1.5.a) enables the SNR degradation from dither to be calculated. Similarly, when that dither is added to the input stream (node B as shown in figure 4.1.5.b) the SNR degradation from dither can be calculated using the STF. Since the addition of dither contributes to the total noise in the output, more NTF suppression may be required in the signal band to satisfy system specifications. This effect can be made small by adding dither which has been pre-filtered to leave insignificant dither noise power in the signal band, so that the re-quantisation noise makes the dominant contribution [VAN89,LIP89] however this may be undesirable in PWM based DACs where high noise levels out-of-band can cause output SNR degradation from sidebands.

Much work has been done to find the required amplitude and characteristics that the dither should have [LIM88,SRI77]. It is suggested in the literature [LIP91] that listeners cannot distinguish correlated noise from uncorrelated noise if its first two moments are not correlated with the signal. Several requirements have to be met, besides satisfying the above constraints for the dither to be considered optimal for a high resolution, low distortion system. These are :

1) the first moment (the mean) of the dither should be zero,

2) the second moment (the variance) of the dither should be small, and

3) the dither PSD should not be large (wrt. signal quantisation), or varying in the signal band.

Maintaining the mean of the dither at zero is a necessary condition for ensuring the linearity of the quantiser, since, if this changes, the system becomes time variant (and possibly amplitude variant). Rectangular PDF dither (RPD) satisfies this requirement but does not have a constant second moment, hence RPD can leave noise modulation which is audibly undesirable. Furthermore, if RPD is used at a size which is not an integer number of LSBs, it no longer linearises [LIP91] because its characteristic function (Sin(x)/x) only becomes zero at integer multiples of the step size.

Since the second moment of the error (the variance) is the dither power, it is important that this be kept small and thus the noise contribution to the output is kept small. One favoured solution to remove noise modulation is to use triangular PDF dither of 2 output LSBs, peak-to-peak amplitude (cf. 1 LSB for RPD). This can be produced by adding (or subtracting) two RPDs and can give zero-mean and minimum-valued, constant variance. By differencing two RPDs an approximately triangular PDF dither can be produced with a high pass characteristic (THPD); this contributes lower noise power to audible frequency bands (see appendix A9 for a PDF and example spectrum). For most noise shapers, 1 bit RPD is adequate since the filtered re-quantisation noise will usually be sufficient to avoid noise modulation except under special conditions (eg. ramp input to a low order sinusoidal shaper). Careful selection of noise sources is still required since even with constant noise power (variance), the spectral proportions (in different frequency bands) must be constant to avoid *subjective* variation.

It should be stressed that such dithers are particularly tailored for audio applications. When dither is used in other applications the above requirements can be revised. Also, alternative dithering strategies exist, notably 'subtractive dithering', whereby dither added at one point in a chain of DSP stages can be subtracted out again later. This can avoid noise degradation from dither completely, while providing dithering's beneficial de-correlating effects; it will be returned to again in section 4.6 .

4.2.4 : Power Gain and Overload Problems With High Order Sinusoidal Shapers.

Examples in section 4.1.3 showed that quantiser overload due to a large amplitude feedback signal presents problems which can only be solved by reducing the input signal range, reducing the output dynamic range (by implicitly scaling), or by introducing distortion (clipping or resetting). This problem becomes increasingly important with higher order sinusoidal shapers since the power gain of H(z) (and hence the NTF) increases approximately exponentially with order [HIO92]. Figure 4.2.4.a below is a table of the first few orders of sinusoidal noise shaper along with their power gain and peak amplitude gain, while figure 4.2.4.b, below that, shows the rapid growth of the sinusoidal family's power gain, with order.

	Coefficients of $H_n(z)$										
order	C1	C2	C2	C3	C4	C5	C6	C7	C8	Max. Gain	Power Gain
1	1	-	-	-	-	-	-	-	-	2	2
2	-2	1	-	-	-	-	-	-	-	4	6
3	3	-3	1	-	-	-	-	-	-	8	20
4	-4	6	-4	1	-	-	-	-	-	16	70
5	5	-10	10	-5	1	-	-	-	-	32	252
6	-6	15	-20	15	-6	1	-	-	-	64	924
7	7	-21	35	-35	21	-7	1	-	-	128	3432
8	-8	28	-56	70	-56	28	-8	1	-	256	12870
9	9	-36	84	-126	126	-84	36	-9	1	512	48620

Figure 4.2.4.a : A Table of Sinusoidal Noise Shaper Characteristics.

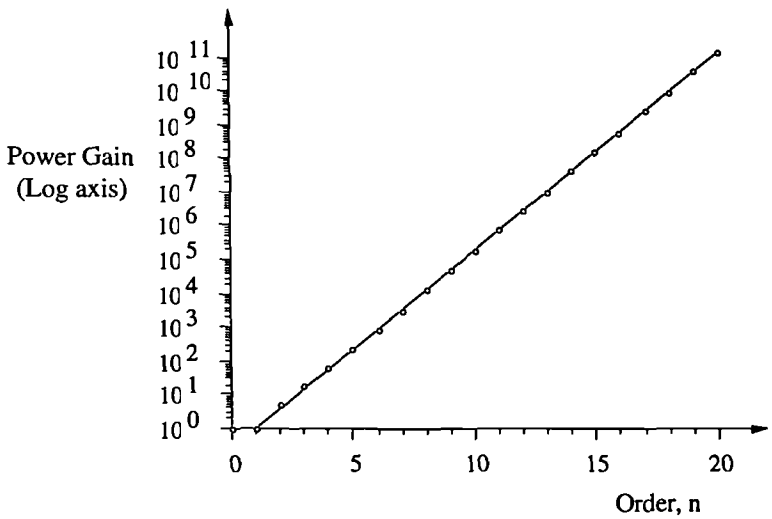


Figure 4.2.4.b : A Graph of Sinusoidal Noise Shaper Power Gain vs. Order.

From this graph, the peak amplitude gain can be seen to increase quickly with order. Most of these filters would limit the usable signal range, and all exhibit highest gain at Fs/2 where the PWM is particularly sensitive (see chapter 2). More sophisticated noise shapers are needed to meet two requirements: (1) to have a lower power gain, and

(2) to have a shape better suited to PWM's characteristics.

Designing for these requirements is the subject of the next section.

4.3 : Design of Arbitrary Shape and Low Power Gain NTFs.

4.3.1 : The Reasons for Using Arbitrary Shape NTFs.

As shown in the last section, using high order sinusoidal noise shapers results in a large noise power gain. This means that the broadband requantisation noise is not only placed at high frequency, but is amplified in the process of doing so, causing problems by limiting the available input signal range (to avoid saturation). When the NTFs are plotted relative to the requantisation noise level, the gain at high frequencies becomes quite obvious (for example: nearly 50 dB higher in the 5 th. order case). If such a filter is used to produce an 8 bit output for example, the worst case error (alternating maximum size) could lead to a total feedback signal of approximately 250 output LSBs, thus requiring almost all the output wordlength for specifying the shaped noise.

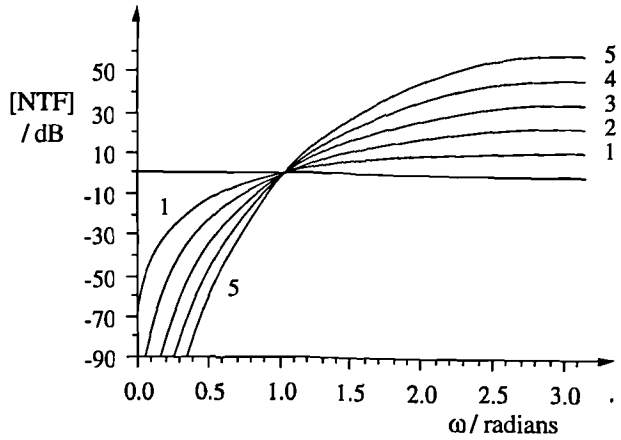


Figure 4.3.1.a : NTFs for Sinusoidal Shapers, Orders 1-5.

Although the signal band requantisation content is increasingly suppressed as the filter order is increased, such filters offer diminishing returns since the inherent quantisation noise from the input cannot be bettered. This 'floor' to the performance that can be achieved suggests that the use of very high orders is not worthwhile. Furthermore, the noise which has been shaped to higher frequencies, can cause problems where it now appears, interfering with any non-linear element which may follow it (eg. a PWM). This is a waste of the information capacity of the channel and can only be alleviated by using a noise transfer function which does not amplify so much at high frequency; such an NTF requires the use of more highly resolved coefficients than are used in the sinusoidal case (ie. 10 bits or more).

In the design of an arbitrary shaped filter, several things have to be considered. First of all, the signal band suppression has to be maintained so that the requantisation level is reduced below the input quantisation level (and by some margin). Secondly, the high frequency bands have to be restricted so that the gain does not become large. Thirdly, the filter has to be scaled so that its first coefficient is unity, allowing the feedback filter H(z) to be calculated from the NTF using equation 4.1.2.e . By using a standard FIR design algorithm (eg. the Parks-McClellan technique, PAR72) an arbitrary shaped filter can be designed, but usually the first coefficient will be less than unity so the filter coefficients should all be scaled up (to make the first, 1):

ie. : $C_i := k.C_i$ $0 \leq i \leq \text{order}, k = 1/C_0$ Equation 4.3.1.a

Scaling shifts the magnitude response so the signal band of the filter is no longer properly specified. This problem can be overcome by iteratively over-designing the NTF until it satisfies the signal band requirements after scaling; high order filters with good spectral performance and low power gain can be found in just this way. An example NTF is shown below to demonstrate the effects of this scaling.

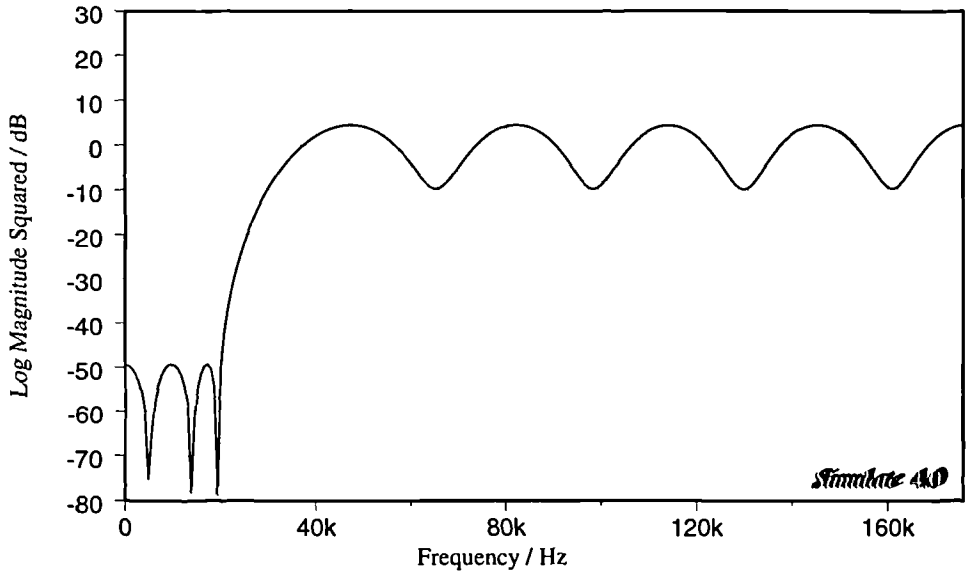


Figure 4.3.1.b : Arbitrary Shaped NTF, First Guess (50 dB Suppression, filter order=25).

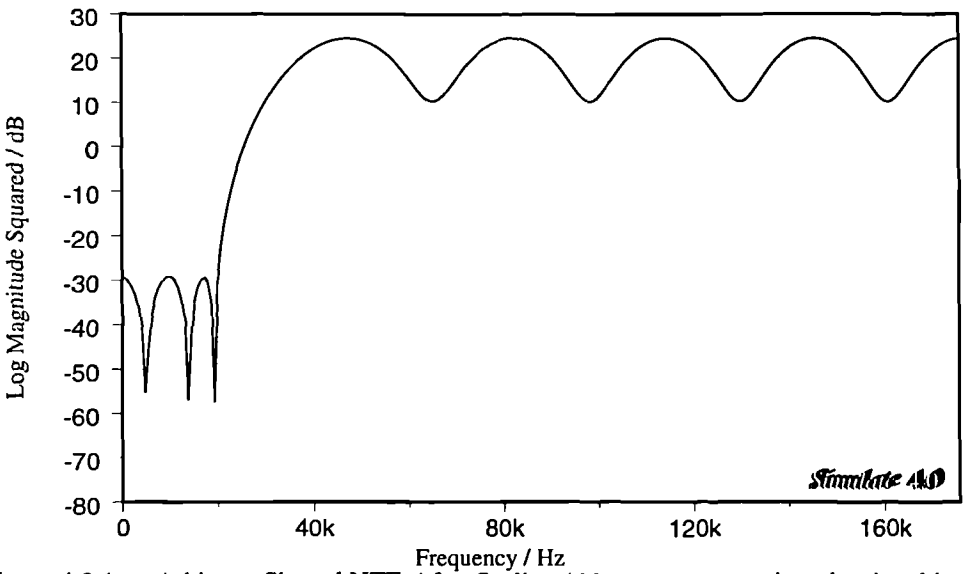


Figure 4.3.1.c : Arbitrary Shaped NTF, After Scaling (this now compromises the signal band).

4.3.2 : Properties of Minimum Gain NTFs.

Although many filters can be constructed with the above shape, only the minimum phase-lag version has the minimum power gain [GER89]. This can be seen as a removal of the redundancy introduced by two excessive constraints: (1) linear phase in the NTF, and (2) limited wordlength coefficients, both of which are a natural consequence of expanding equation 4.2.1.a in the sinusoidal case. NTFs with the minimum phase-lag property will be referred to simply as 'minimum phase' and noise shapers based on such filters will be called 'minimum phase noise shapers' (MP-NS).

While MP-NS do not usually have integer coefficients, similar, minimum phase NTFs can be found which have near minimum power gain (for a given shape) and can be implemented using restricted wordlength processing. One useful consequence of converting a linear phase NTF to minimum phase is that the order of magnitude of its coefficients become more similar enabling the use of alternative structures in an implementation to avoid internal overflow in the filter, H(z) as will be shown in section 4.4.3.

Converting a linear phase NTF to minimum phase does not alter the magnitude response of the filter (unlike scaling the coefficients), and consequently the peak magnitude gain remains the same. In terms of coefficients, the sum of the coefficient magnitudes remains unaltered so the sum of the squares of the coefficients also remains the same. This is a significant point because the danger of overflow (saturating the quantiser) is still present. Furthermore, this process does not produce a filter with a first coefficient of unity. However, after scaling has been applied to correct for this, a new minimum phase filter with lower peak magnitude gain is generated. This scaling is a shift on the logarithmic magnitude frequency response to the point at which the integral of the logarithm of the magnitude response over all frequencies is zero [GER89]; with knowledge of the ratios of bandwidths of the signal and noise bands, an estimate of the required over-design at the linear phase stage can be provided since :

$$\int_0^{\pi} \{ \mathrm{Log} \, | \, NTF(\omega) \, | \, \} \, d\omega \geq 0$$

Equation 4.3.2.a

Put graphically, the areas A, and B (as shown in figure 4.3.2.a) are the same:

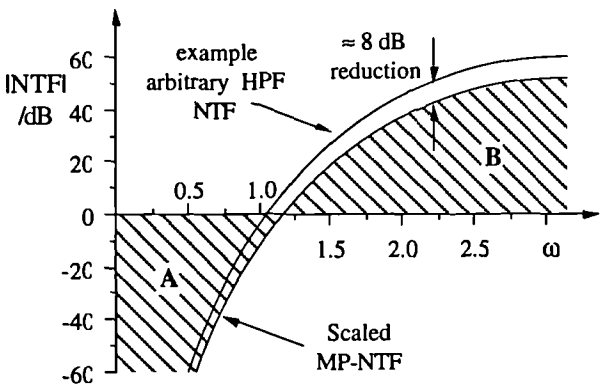


Figure 4.3.2.a : Fifth order Arbitrary NTF, converted to Minimum Phase.

This is of particular interest because this bears out the fundamental rule that the information capacity of the channel cannot be increased by processing (without prior knowledge of the signal). In essence, the best that a noise shaper can achieve with a given shape of NTF is to preserve the information capacity of the output by simply moving the band in which the noise lies.

Maintaining a zero valued integral for the logarithm of the magnitude response reveals one important point about the shape of the filter, namely that excess stopband suppression (in the low frequencies for conventional noise shapers) will demand excess high frequency gain. Thus it becomes of paramount importance not to over-design the stopband suppression requirement (as was common in the

sinusoidal case). It is still important to remember that although the scaled, minimum-phase NTF has low power gain, this gain is associated with the filter's shape. A filter with a slow rate of change of magnitude response in the transition band will unnecessarily suppress frequencies beyond the signal band. This leads to the power gain of the filter always being greater than one (as can be seen from the coefficients, where the first has a value of unity and the rest must have a value to provide some effect). Further room for improvement can be expected if an ideal shape (ie. NTF magnitude response) can be found - that shape depending on six things : the oversampling ratio, the number of bits dropped and type of quantisation, the input quantisation noise 'target', the order of filter that can be used, and its coefficients' wordlength. Evaluating what the NTF's signal band requirements are (ie. the requantisation noise stopband) is the subject of the next section.

4.3.3 : Evaluating Arbitrary Shape NTF Signal Band Requirements.

The signal bandwidth and resolution can be calculated for a given desired output SNR, once the input wordlength, the output wordlength and the oversampling ratio have been defined, assuming filter coefficient wordlengths will not limit the system. Noise shaping always adds to the noise in the signal band, but this can be made very small by increasing the stopband suppression to beyond the quantisation level of the input. The gain of the NTF in the signal band is calculated from knowing the allowable requantisation noise content of the desired SNR (since the input quantisation noise is known). As discussed above, the output cannot have a higher SNR than the input, in any part of the band. Given that the output SNR is not as good as the input's, (evaluated across the signal band) the lower the desired output SNR, the better the NTF's signal band suppression has to be; also, as the desired output SNR approaches ideal (the input SNR) the NTF signal suppression increasingly has to be improved, to meet the desired performance.

From the output wordlength the requantisation noise power can be calculated, but this does depend on the type of quantiser used. If it can be assumed that the distribution of quantised signal is rectangular, the noise power can be calculated in the rounding and truncating cases by integrating the squared error. In the truncating case the error varies linearly from 0 to the quantisation step size, Δ, in the rounding case this is from - $\Delta/2$ to $+\Delta/2$:

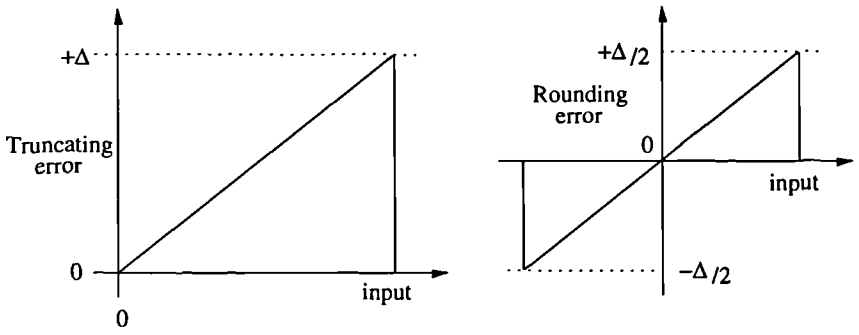


Figure 4.3.3.a : Quantisation Error Power for Truncation and Rounding.

Integrating the squared quantisation error for the two cases yields a rounding error power of $\Delta^2/2$ whereas the truncating error power is Δ^2. In short, the requantisation noise power by rounding will be 3 dB lower than that from truncating in the quantiser. It should also be remembered that

designing with truncating quantisers leads to purely positive valued error in the feedback loop to the filter H(z). For the same reasons, the rounding quantiser will produce the same size maximum positive and negative values of the error signal. These two features permit different implementation simplification.

From the NTF signal band gain, an estimate of the ratios between the minimum and maximum magnitude gains can be calculated for a chosen transition bandwidth (obtained by equating areas); this can then be used in a standard FIR digital filter design programme. The resulting filter can be converted to minimum phase and scaled to make the first coefficient unity, and from this NTF the filter coefficients for H(z) follow. The stopband suppression can be found as follows [GOL92] :

Four parameters are required to calculate the stopband suppression:

1) the desired output SNR,

2) the input wordlength, b bits,

3) output wordlength, b' bits, and

4) the oversampling ratio, L.

In general the signal to noise ratio is defined as,

$$\cdot SNR = 10 . Log_{10} \left[\frac{\text{SIGNAL POWER}}{\text{TOTAL NOISE POWER}} \right] = \frac{\sigma_x^2}{\sigma_{qn}^2 + \sigma_{rn}^2}$$

Equation 4.3.3.a

where σ^2_x is the signal power,

σ^2_{qn} is the input quantisation noise in the signal band, and

σ^2_{rn} is the requantisation noise added to the signal band.

From equation 4.3.2.a, the allowed requantisation noise can be expressed as,

$$\sigma_{rn}^2 = \sigma_x^2 . 10^{(-SNR/10)} - \sigma_{qn}^2$$

Equation 4.3.3.b

Assuming the signal to be a sinusoid, of amplitude X, the power in it is:

$$\sigma_x^2 = \frac{X^2}{2} \quad \text{for the signal : } X.Sin(\omega t)$$

Equation 4.3.3.c

and assuming the signal has been interpolated so all the quantisation power is in the band of interest,

$$\sigma_{qn}^2 = \frac{\Delta^2}{12} \quad \text{where } \Delta = \frac{X}{2^{b-1}}$$

Equation 4.3.3.d

which expressed in terms of the signal's amplitude, can be rewritten,

$$\sigma_{qn}^2 = \frac{X^2}{12 . 2^{2(b-1)}}$$

Equation 4.3.3.e

Knowing the output requantisation noise contribution to be over the band 0 - π/L, its power is:

$$\sigma_{rn}^2 = \frac{1}{\pi} \int_0^{\pi/L} \{ PSD_{rn}(\omega) \}\, d\omega$$

Equation 4.3.3.f

but the PSD of the shaped requantisation noise in the signal band is simply the PSD of the unshaped requantisation noise multiplied by the gain of the NTF ($0 < \omega < \pi/L$), 'G'.
The unshaped requantisation noise power can be calculated from the number of bits in the output, b',

$$\sigma_q^2 = \frac{1}{\pi} \int_0^{\pi} \{ PSD_q(\omega) \}\, d\omega = \frac{X^2}{12.2^{2(b'-1)}}$$

Equation 4.3.3.g

so the truncating requantisation noise can be expressed as,

$$\sigma_{rn}^2 = \frac{1}{\pi} \int_0^{\pi/L} \{ \sigma_q^2 . G^2 \}\, d\omega = \frac{1}{L} . \frac{X^2}{12.2^{2(b'-1)}} . G^2$$

Equation 4.3.3.h

which substituting back into 4.3.2.b, using 4.3.2.c and 4.3.2.e yields,

$$G^2 = \left\{ 6.10^{\left(-\frac{SNR}{10}\right)} - 2^{2(1-b)} \right\} . L . 2^{2(b'-1)}$$

Equation 4.3.3.i

This result is not only useful for determining the stopband requirements for filter design programmes, but can be used to define the error margins in optimisation programmes. This technique is examined in section 4.4 and has been found to be most satisfactory.

4.3.4 : Minimum Delay NTFs and Their Relationship to MP-NS.

Referring back to section 4.1.1, the characteristics of noise shaping was approached from a time domain point of view and this again becomes useful in understanding why minimum phase NTFs are so useful in minimising the output noise power.

If the noise shaper is considered to be a modified negative feedback network, being upset by an imperfect forward gain function it is easy to see that the sooner the input can be modified to take into account the output errors, the closer the input waveform will be followed. In short, the faster the feedback network passes the error signal to correct subsequent output, the smaller the deviation in the output from the intended signal (the input).

Minimum phase-lag filters are particularly interesting in this application, since for a given filter shape they represent an optimum reduction in the delay between the output and the input (expressed as phase-lag). Since the noise power gain is the square of the output deviation from the input signal, minimising the delay of the NTF will minimise the noise power gain of the noise shaper.

The minimum delay criterion becomes even more revealing when filters of different signal bandwidth are compared and it is noticeable that coefficients of low index (ie. C_1,C_2) in the feedback filter become increasingly significant for a larger signal band (as a fraction of the sampling frequency).

Scaling the NTF to ensure a first coefficient of unity, does not affect its delay (just its gain), so delay minimisation can be applied to noise shapers to achieve a reduction in the error size (ie. the noise power gain). Furthermore, when only some of the frequency response of the NTF is specified and the remainder is arbitrary, the shape of the unspecified portion of its frequency response can determined for the minimum gain case by constructing the minimum delay shape.

Evaluating the shape for minimum delay is not easy, but can be estimated knowing that the magnitude response and the delay response are a Hilbert transform pair for a minimum phase filter. This proposal is confirmed by the fact that if the group delay of the filter is minimised, the integral of the group delay will also be minimised, and this is the phase. In other words, the minimum delay shape will implicitly yield a minimum phase-lag response. Furthermore, if the error power can be spread both *before* the current sample, as well as afterwards (recursively), the error power could be yet smaller. To verify this a feedforward network can be used as below (although this will corrupt the nulling of the LSBs in the output produced by the quantiser).

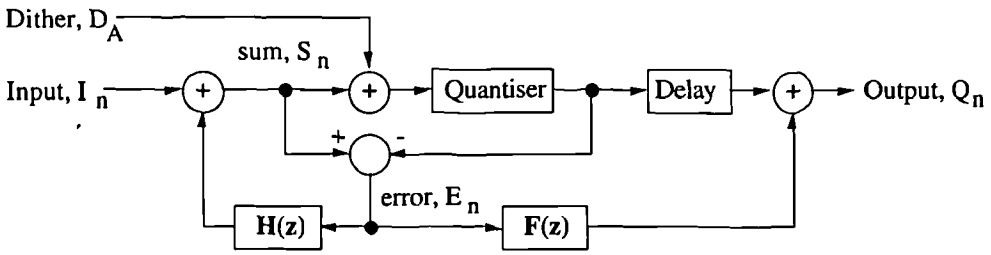


Figure 4.3.4.a : Feedforward Filtering to Demonstrate Low Power Gain Noise Shaping.

If the filtering is applied in this way, the error can be spread symmetrically on neighbouring samples, leading to a symmetric NTF impulse response (and hence linear phase). In this case, delay minimisation leads to a filter with the largest coefficients grouped near the intended sample, similar to the conditions required for minimum inter-symbol interference filters . This demonstrates that minimum phase-lag designs are only special case of minimum delay filters, specifically for 'one sided' or purely recursive designs. The spectral performance of this system and a purely recursive noise shaper are shown below to demonstrate the reduction in noise power-gain that can be achieved.

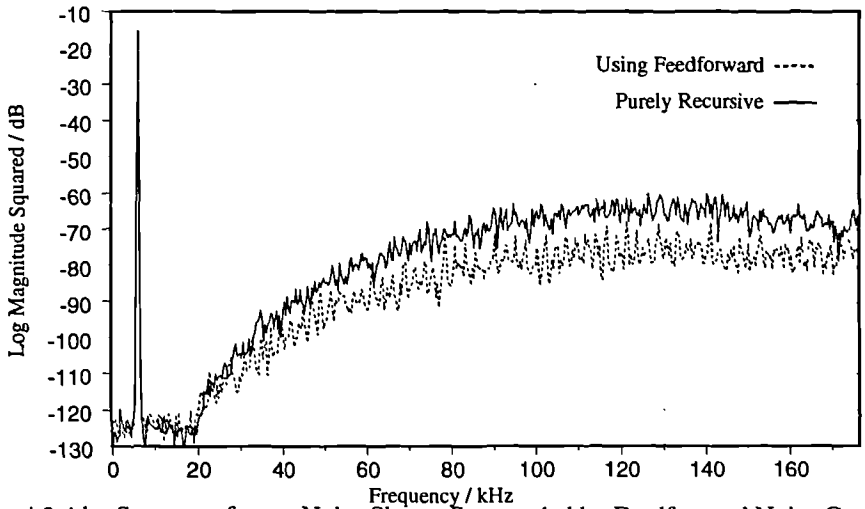


Figure 4.3.4.b : Spectrum from a Noise Shaper Surrounded by Feedforward Noise Correction.

In a practical noise shaper application, the LSBs of the output must be nulled for the system to be used as a way of reducing the wordlength of an oversampled signal. Feedforward noise correction cannot be applied in the way shown in figure 4.3.4.a because the feedforward filter will produce finely quantised (if not full precision) error correction signals to add to the quantised signal, thus destroying the wordlength reduction that has been achieved. However, if the delay line is placed ahead of the noise shaper, as shown in figure 4.3.4.c, filtering of this kind can be applied under certain conditions.

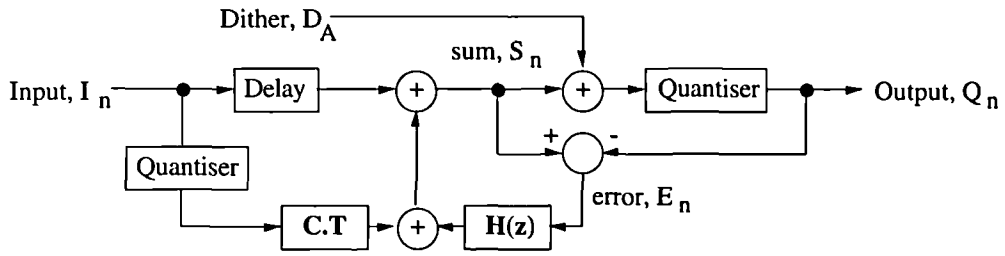


Figure 4.3.4.c : Feedforward Filtering Using a Look-up or 'Correction' Table.

Difficulties arise in applying the feedforward filter coefficients, since the filtered error on which the quantiser input depends, is itself affected by the quantiser output. There are two solutions to this problem : either the filtered error should be corrected iteratively (preferably precalculated) or else the feedforward error should produce integeric output.

Considering the first solution, as shown in figure 4.3.4.c, the feedforward filter has been replaced by a correction table because the filtering can no longer be applied in a conventional digital filter structure. To demonstrate what the required contents of this table are a simple example will be used : If the overall NTF has coefficients $H(z^1)=0.5$, $H(z^{-0})=-1$,$H(z^{-1})=0.5$ then for an underlying current error e_0, a 'previous' error of e_{-1}, and a next error value of e_{+1}, the iterating error can be seen to settle as shown below in figure 4.3.4.d .

current error	$-e_{-2}$	$-e_{-1}$	$-e_0$	$-e_{+1}$	$-e_{+2}$
1 st prediction	$+0.5e_{-3}$	$+0.5e_{-2}$	$+0.5e_{-1}$	$+0.5e_0$	$+0.5e_{+1}$
1 st recursion	$+0.5e_{-1}$	$+0.5e_0$	$+0.5e_{+1}$	$+0.5e_{+2}$	$+0.5e_{+3}$
2 nd prediction	$+0.25e_{-4}$	$+0.25e_{-3}$	$+0.25e_{-2}$	$+0.25e_{-1}$	$+0.25e_0$
	$+0.25e_{-2}$	$+0.25e_{-1}$	$+0.25e_0$	$+0.25e_{+1}$	$+0.25e_{+2}$
2 nd recursion	$+0.25e_{-2}$	$+0.25e_{-1}$	$+0.25e_0$	$+0.25e_{+1}$	$+0.25e_{+2}$
	$0.25e_0$	$+0.25e_{+1}$	$+0.25e_{+2}$	$+0.25e_{+3}$	$+0.25e_{+4}$
	⋮	⋮	⋮	⋮	⋮

Figure 4.3.4.d : A Table Showing the Successive Approximation of the Feedforward Filtering.

Putting this in a more general form, with a feedforward coefficient of 'j' and a feedback coefficient of 'k', the current error, e_0, can be seen to be replaced by :

$$
\begin{aligned}
&\ldots\ldots \\
&+ k^2 * e_{+2} * (1 + 2jk + 2j^2k^2 + 2j^3k^3 + \ldots\ldots) \\
&+ k * e_{+1} * (1 + 2jk + 2j^2k^2 + 2j^3k^3 + \ldots\ldots) \\
e_0 := &+ e_0 * (1 + 2jk + 2j^2k^2 + 2j^3k^3 + \ldots\ldots) \\
&+ j * e_{-1} * (1 + 2jk + 2j^2k^2 + 2j^3k^3 + \ldots\ldots) \\
&+ j^2 * e_{-2} * (1 + 2jk + 2j^2k^2 + 2j^3k^3 + \ldots\ldots) \\
&\ldots\ldots
\end{aligned}
$$

Equation 4.3.4.a

From this, it should be noted that the product of the feedforward and feedback coefficients must be less than unity for the series to converge. Fortunately, this is one of the characteristics which has been observed in NTFs designed for minimum delay. Having evaluated this table (which can be considerably more complicated for higher order filters), the relationship between each sequence of quantised inputs and the set of required outputs can be calculated, but only if a sufficient number of terms is considered. Since the contributions from adjacent samples decays as $1/j^n$ and $1/k^n$, the number of adjacent samples to consider can be found, once the coefficients have been chosen, and the allowable error in the table is known (dependent on the passband suppression of the filter).

The second technique for handling the error feedforward filter is considerably simpler but only approximate. By designing filters whose 'current' and feedforward taps are integers, and testing the difference between the quantised input and the quantised output (ie.: approximately measuring the noise signal amplitude), an anti-phase signal can be added into the stream to re-distribute the peak noise in advance of the sample on which it occurs. A block diagram of such a system is shown below :

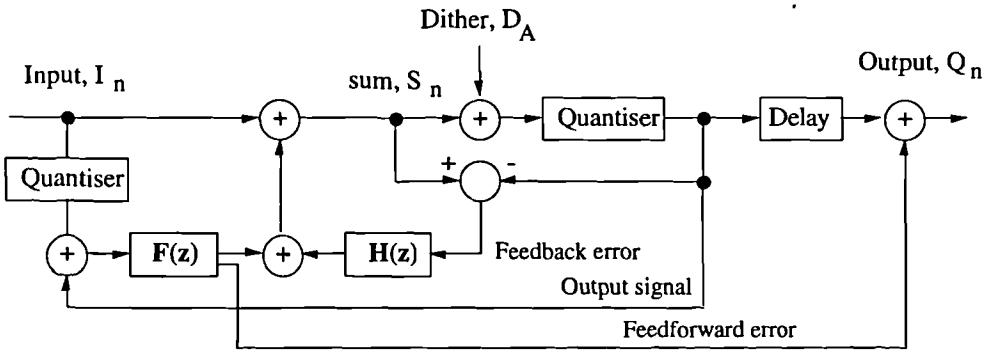


Figure 4.3.4.e : Estimated Feedforward Error Correction for Low Gain Noise Shaping.

The requirement that the 'current' tap is an integer means that when a correction signal is added in anti-phase, it will reduce the output sample value without altering the current error value. Similarly, the integeric feedforward taps permit feedforward error signals to be added to the output without destroying the nulling of the output LSBs achieved by the quantiser. Such integeric coefficients limit the usefulness of this kind of filtering because they approximate low gain filters very badly. Only one or two taps have been successfully implemented in the feedforward section of filter 'F' (see figure 4.3.4.e), but low gain NTFs in the feedforward filter can still be achieved. The spectral performance of this system and the best performance of a purely recursive noise shaper of the same order are shown below, from which the lower output noise power in the more symmetrically corrected noise shaper can be seen (ie. in the estimated feedforward error correction model).

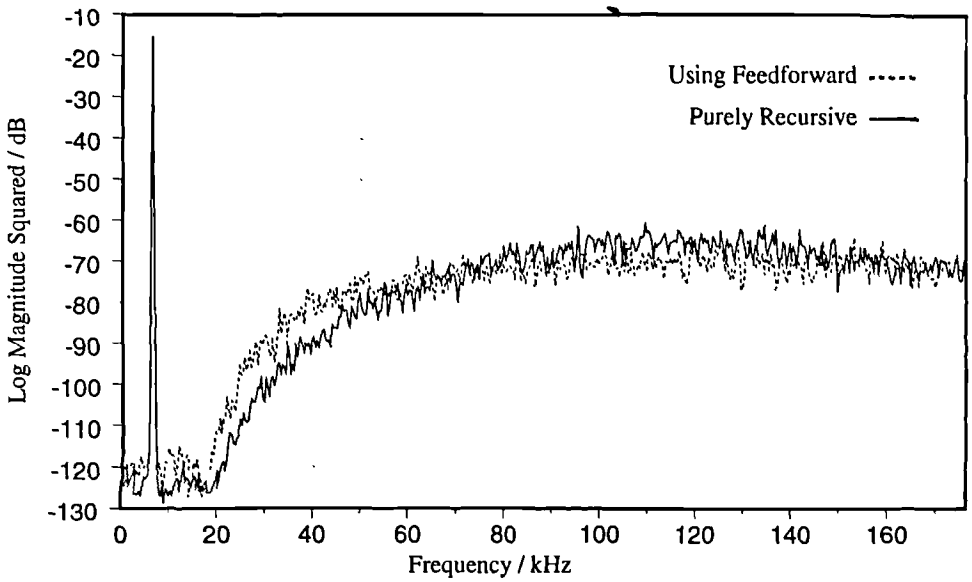


Figure 4.3.4.f : Purely Recursive, and Estimated-Error Feedforward Correction Noise Shaper Spectra.

4.3.5 : Performance of Minimum Phase Designs vs. Order.

The order of minimum phase NTFs has to be found empirically since their shapes are arbitrary and dependent on system requirements (signal band performance, noise power gain, importance of some frequency bands) rather than predicted by an equation as in the sinusoidal case (cf. equation 4.2.1.a).

As with linear phase designs, the order is a function of the transition bandwidth, and the allowed passband and stopband ripple. Low order designs exhibit larger passband ripple as a consequence of constraining the stopband requirements to satisfy the output noise power requirements defined by the input/output wordlength specifications. With too low an order these specifications may be impossible to meet, especially for very high resolution systems or very low oversampled systems. As the order is increased, the power gain can be reduced since the transition band can be more tightly controlled, reducing the amount of area 'A' (see figure 4.3.2.a) by changing it from a trapezium to nearer to a rectangle. In a similar way, area 'B' will be increased, and then the entire NTF can be shifted downward (scaled) to form a new balance, and a new minimum phase NTF.

If the narrowing of the passband is carried on indefinitely, a limit is reached where the two areas 'A' and 'B' are rectangles, and the power gain is reduced to unity; this represents the theoretically optimal noise shaper (in a minimum power gain sense). This limit is not practical in itself since it would require a filter of infinite order, but it can be used to establish a lower bound on the resolution that can be achieved for a given oversampling ratio.

To demonstrate this point, consider a four times oversampled system. This will have a signal band in the NTF which occupies $\pi/4$ radians, leaving $3\pi/4$ radians for noise. This implies that the area of noise gain will have one third of the height compared to that which the area of noise suppression has in depth (each measured in decibels), as shown below:

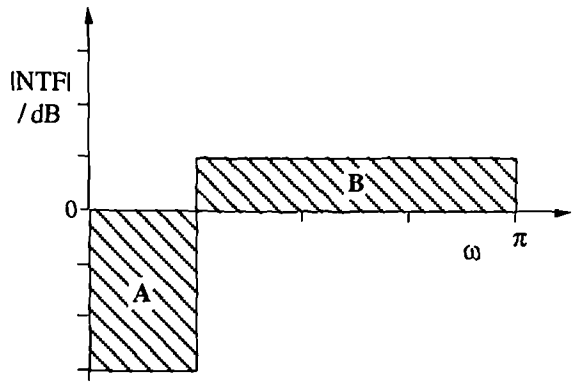


Figure 4.3.5.a : The Limit of Noise Shaper Operation.

When a noise shaper approaches these conditions there is very little redundancy in the design. Coefficient accuracy has to be very high and system complexity is enormous. In general orders from five to twenty have been found to be useful, yielding noise power gains between 2 and 5. A set of example filters offering reduction by 12,13,14,15 & 16 bits (input to output) with 16 times oversampling have been found using techniques discussed in the next main section (4.4). These have been plotted below (figure 4.3.5.b) to show the asymptotic behaviour of the power gain as a function of the order of the filter.

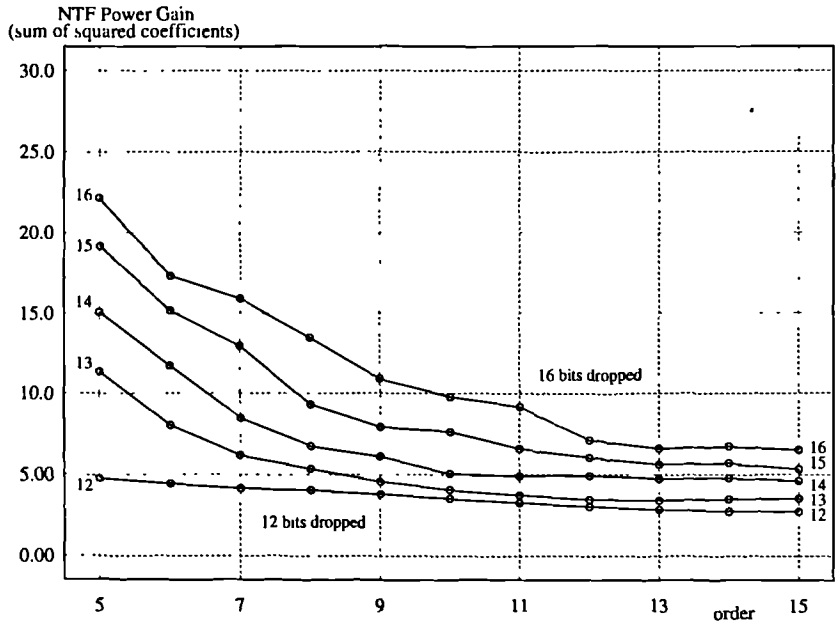


Figure 4.3.5.b : Power Gain of NTFs vs. Order, Each With The Same Signal Band Performance.
(nb.: the asymptotic behaviour of this function, suggesting little potential for improvement with order)

4.3.6 : Conversion of a Linear Phase NTF to its Minimum Phase Equivalent.

As discussed in section 4.3.1, the filter required for an arbitrary shaped NTF in a noise shaper can be designed using a standard linear phase algorithm, and scaled such that its first coefficient equals 1 (equation 4.3.1.a). In section 4.3.2, the lowest gain version of a filter for use in purely recursive noise shaping was seen to be a minimum phase-lag one. One systematic approach to designing low gain noise shapers is to convert a linear phase filter to minimum-phase, and then scale it. The technique for converting linear-phase sequences to minimum-phase equivalents is the subject of this section.

The linear phase sequence represented by the coefficients of the NTF (samples of the impulse response) can be viewed as the sum of a minimum phase part and a maximum phase part [OPP75]. To calculate the minimum phase part, the real part of the complex cepstrum of the linear phase sequence has to be isolated, and the impulse response regenerated from this having forced symmetry on the complex cepstrum. This can be done because the logarithm of the magnitude of the Fourier transform will exhibit symmetry about the centre of the sequence if the sequence was minimum phase, so the right hand half can be constructed from the left hand half and thus a periodic approximation to the complex cepstrum can be produced (eg. by IDFT). The minimum-phase sequence can then extracted from the complex cepstrum, and the NTF's impulse response can be calculated. This process is made particularly simple, because the IDFT which would normally be required can be replaced by the DFT since the complex conjugate of a logarithm is the same as the logarithm itself (purely real). This process was coded in FORTRAN as programme, 'mpc.for' (see appendix A.3.2).

Care has to be taken in this process to avoid time aliasing since a finite length DFT has to be used so the sequence should be fully contained within the left half of the cepstrum for a periodic sequence to be generated without significant error. Use of a discrete Fourier transform of at least five times the sequence length has been found to give satisfactory results. This process relies on the Fourier transform being defined at all the discrete points in the DFT used to derive the frequency response. In particular, zeros in the transform may lead to large inaccuracies and it is suggested that points falling exactly on a zero should either be replaced by a small value [BOI81] or a new length of DFT should be chosen to avoid the zero.

The operation of 'mpc.for' can be checked by plotting the zeros of the two sequences and ensuring that all zeros outside the unit circle (on the z plane) are reflected to inside the unit circle while the magnitude response of the sequence remains unchanged. Programme 'laguerre.for' (see appendix A3) can be used to find the zeros of the sequences which can then be conveniently displayed. Alternatively, 'laguerre.for' can be used to find the roots (zeros) of the polynomial (the series) and after conversion to polar form, the reciprocals of the zeros with radius larger than one can be found and multiplied through to form a new (minimum phase) sequence. Shown below is an example of what can be achieved with either of these techniques, working with a low pass filter. Filters for use with noise shaping will commonly be high pass, for which the same technique applies.

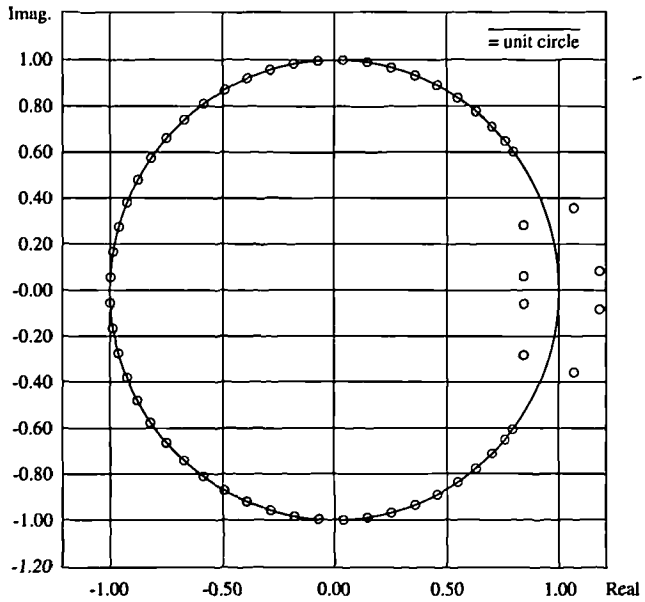


Figure 4.3.6.a ; Filter Zeros On The Complex-Z Plane Before Conversion to Minimum Phase (nb.; zeros outside the unit circle).

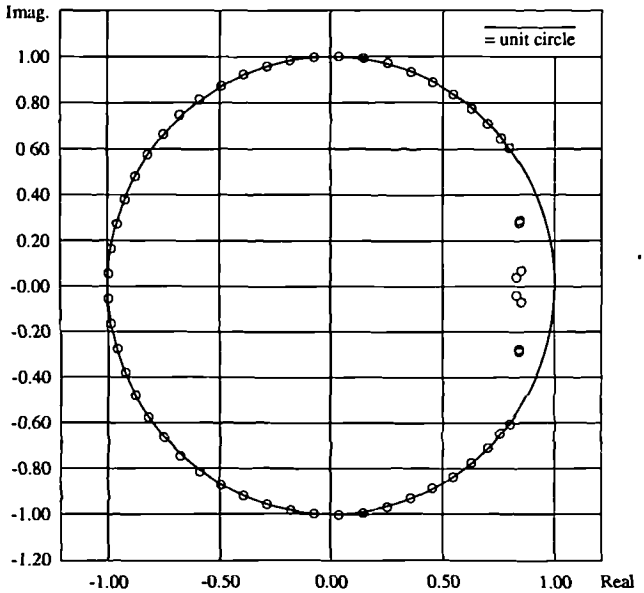


Figure 4.3.6.b ; Filter Zeros On The Complex-Z Plane After Conversion to Minimum Phase (nb.; zeros now reflected-in).

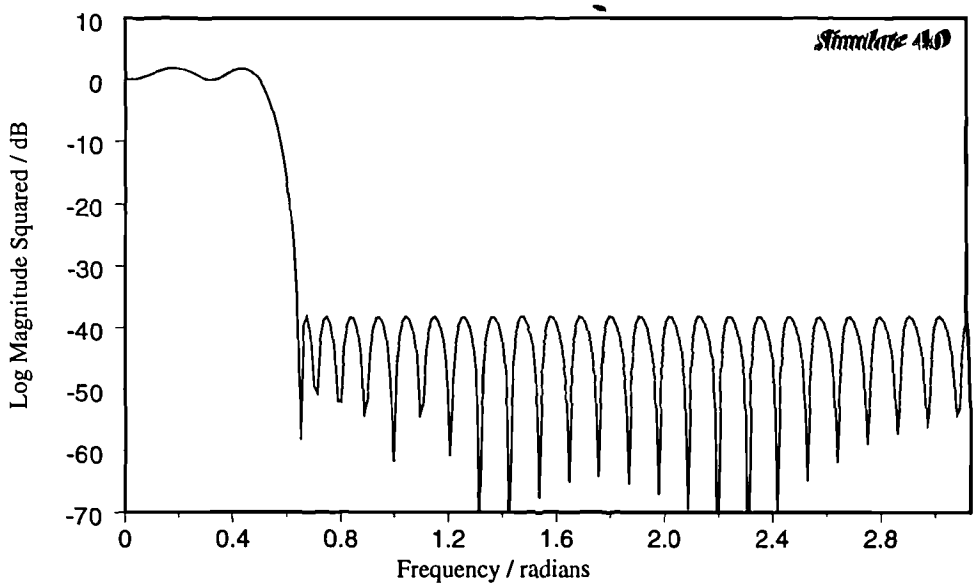


Figure 4.3.6.c : Filter Frequency Response Before Conversion to Minimum Phase.

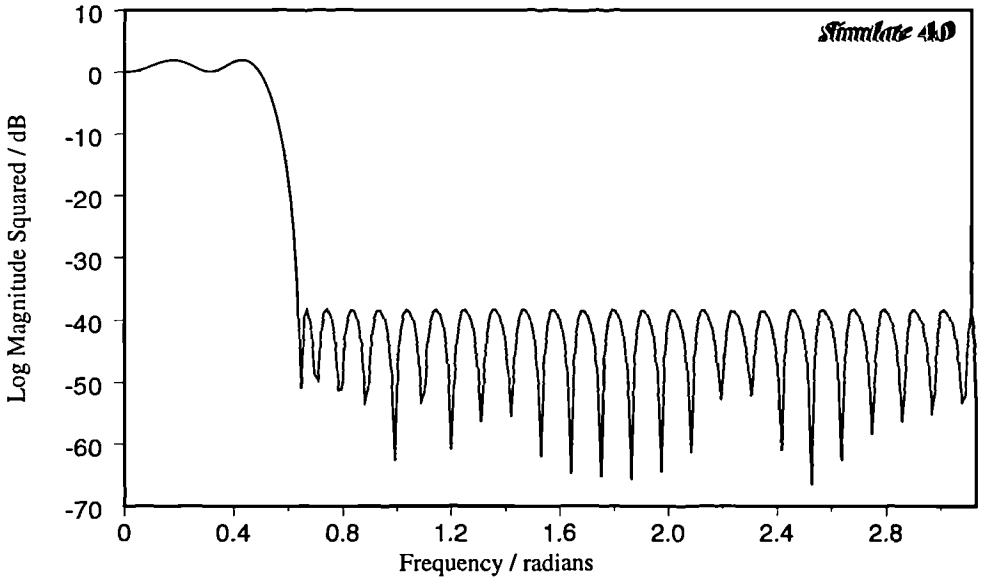


Figure 4.3.6.d : Filter Frequency Response After Conversion to Minimum Phase.
(nb.: the magnitude response remains unaltered)

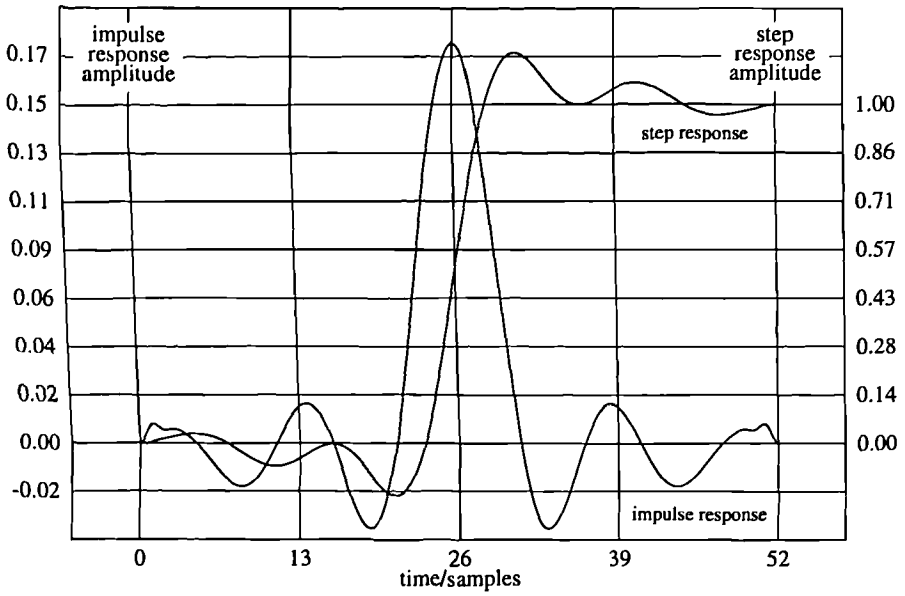


Figure 4.3.6.e : Filter Impulse and Step Responses Before Conversion to Minimum Phase.

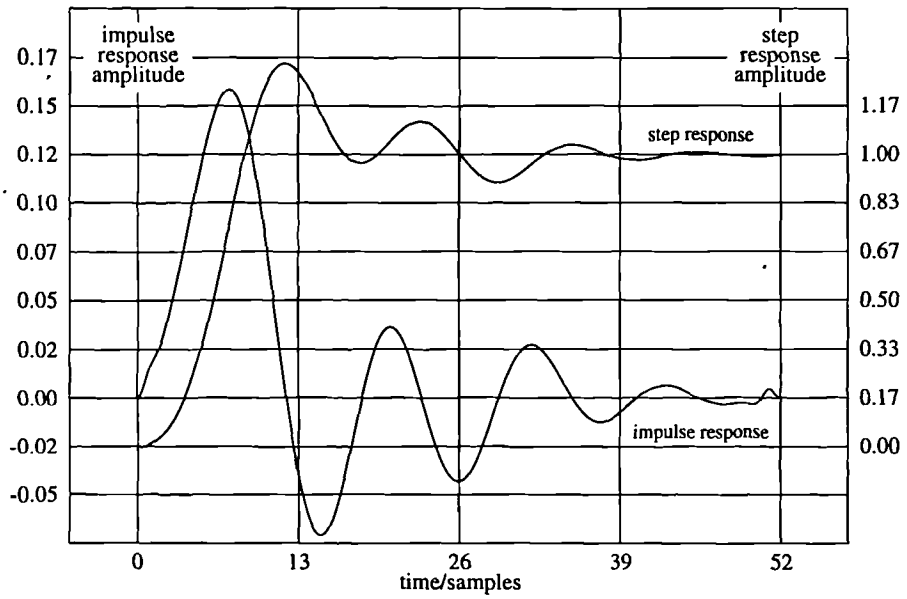


Figure 4.3.6.f : Filter Impulse and Step Responses After Conversion to Minimum Phase.
(nb.: the input energy - the impulse - is now conveyed to the output, far earlier than before)

4.4 : Special Design Techniques.

4.4.1 : Simplex Optimisation of NTFs For Minimum Gain.

Designing NTFs using standard FIR filter design programmes is iterative, involving estimation of the requirements (relying on a guess of an acceptable transition bandwidth) as well as subsequent conversion and scaling in every cycle. While acceptable NTFs can be designed in this way, it is likely that a higher order than is absolutely necessary may be chosen to avoid excessive design time. A good solution to finding more efficient NTFs (ie. lower order for given gain, or lower gain for given order) is to apply the known stopband requirements as an error criterion in an optimisation routine reducing the power gain of the NTF. Using the simplex optimisation routine [NEL64] filters of low order (ie. <20) can be found in a matter of minutes with power gain between two and ten times lower than with traditional techniques. The operation of the basic simplex was modified to speed up its operation by automatic extension along the search surface in the direction of successful 'moves', and contraction in areas of little success.

The optimiser works by taking an initial guess and generating from it a group of guesses by scaling each coefficient in turn by a preset factor (eg. 1.1). This group of points (or vertices) form 'the current simplex' which can be visualised as a group of points on a 'hilly' multi-dimensional surface. The dimension of the surface is simply the number of coefficients to be optimised, using each axis to represent one coefficient, and the optimisation routine tries to move 'downhill' towards the nearest minimum. If it is to gain 'momentum' in some direction it may well pass small features in the surface. If the current simplex comes to the bottom of a substantial 'hollow', it may well settle there and when successive moves are not far from the last, the coefficient values are taken from the simplex's position along each axis. The height of each point is an analogy of the 'error' that its ordinates imply when evaluated in a user defined error function. An example 'contour map' of moves is shown below for a two parameter (x,y) example.

Start with point 1
Generate point 2 & 3
(by multiplying up 1's coordinates)
reflect the worst of 1,2,3 through their centroid
Is the new point (4) any better ? (YES)
If reflected further (5) is this better ? (YES)
OK, reflect the worst of 2,3 & 5
Is this (6) better ? (YES)
Can it be reflected further (NO)
OK, reflect the worst of 3,5, & 6
Is this (7) better (YES)
Can it be reflected further (NO)
OK, continue with 5,6 &7...

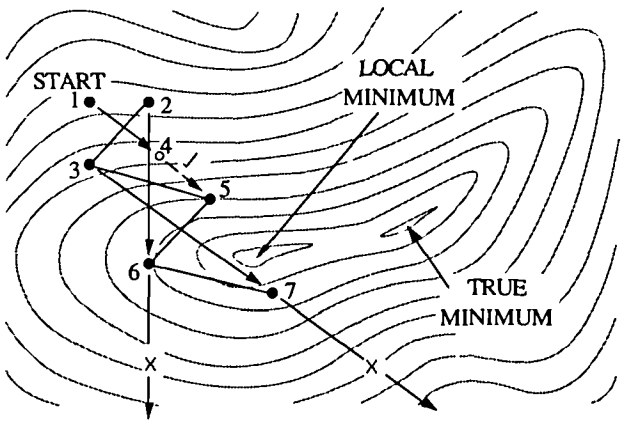


Figure 4.4.1.a : Map of Moves in a Simplex Optimiser

To move the simplex appropriately, all the vertices of the current set are evaluated, and the worst and best values are retained for comparison. The centroid of all the points in the simplex except the one with maximum error is calculated, and the vertex with worst error is reflected through the

centroid. If this new point has a lower error (height) than the best, a another point is tried looking further in the successful direction by an expansion factor (eg. 150%); this is repeated until the error stops improving. The best point so found replaces the old worst point and a new simplex is formed. If the attempted new point is worse that the current worse in the group, it is not accepted and a repeated contraction factors is applied to the reflection (eg. 50%) until a point better than the old worst error point is found. If the initial new point is found to lie between the current worst and best, it is accepted in place of the worst and the new simplex proceeds directly. When the distance between the vertices in the current simplex are less than a user defined 'acceptable' minimum, the optimisation stops. Since the simplex technique can stop in a local minimum, the routine is repeatedly restarted with the best point as an initial guess until no better minima are found.

The optimisation programme, 'nsopt.for', was written in FORTRAN, and a flow diagram of its operation is shown below:

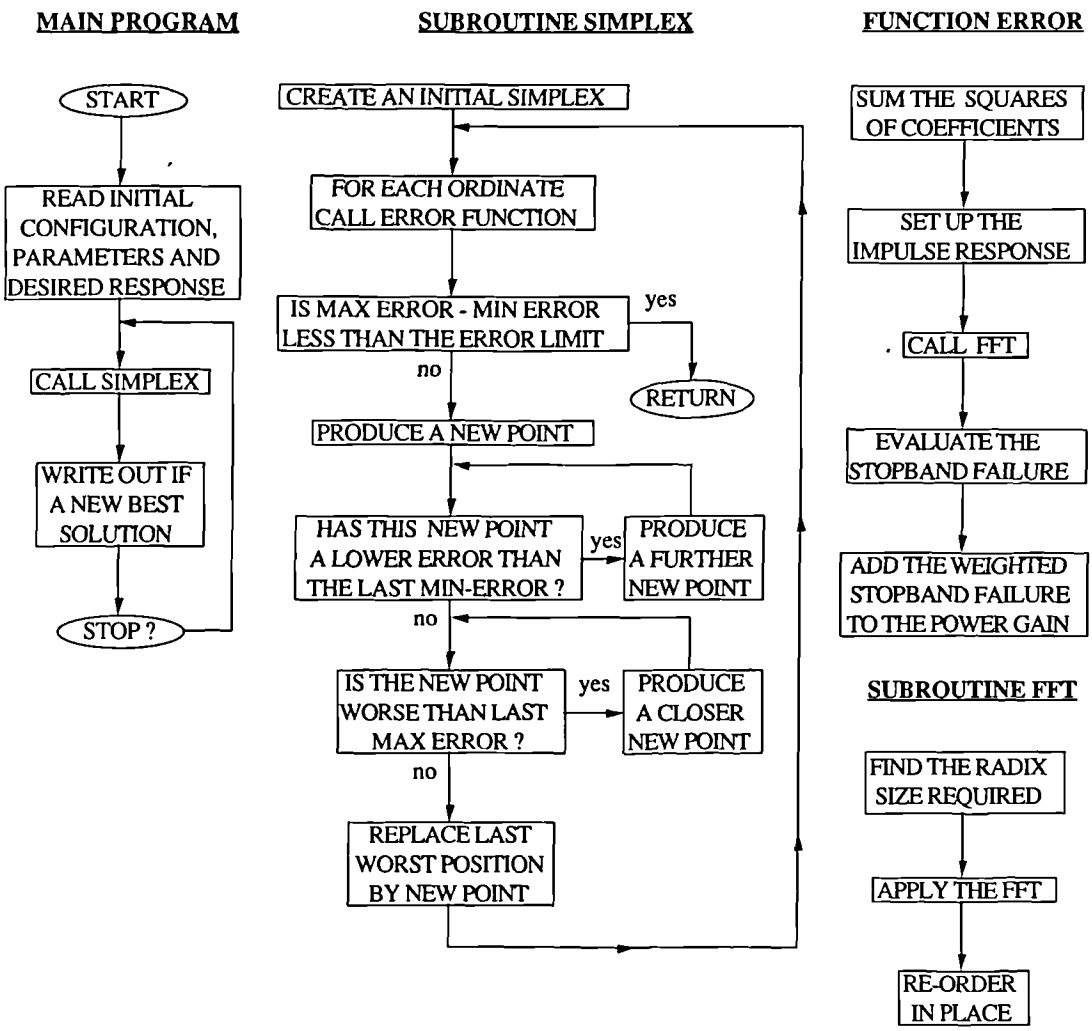


Figure 4.4.1.b : Flow Diagram of the Operation of 'NSOPT'

The user-defined error function is written to return a larger error for increased power gain or increased failure to satisfy the signal band requirements. The power gain can be calculated by summing the square of the trial coefficients and the stopband failure can be measured as the sum of the square of

normalised frequency response deviations *above* the required stopband level. A standard FFT was used to determine the magnitude of the frequency response, checking at discrete frequencies in the stopband; values of the response found above the stopband requirement are normalised with respect to the required stopband level, squared (to emphasise gross failure), and then accumulated. The total stopband failure is subsequently weighted to give it appropriate priority over the power gain (ten to one hundred thousand times more important), before being added to the power gain to form the total 'error'. A segment of source code for this function is shown below:

```
      DOUBLE  PRECISION  FUNCTION  ERROR(FAC)
      EXTERNAL  FFT,  R8TR,  ORD1,  ORD2
      COMMON  /EF/  NOP,  NOS,  C,  N,  NOB,  DELTA,  WEIGHT
      INTEGER  NOB(514)
      DOUBLE  PRECISION  S,  ERR,  A(514),  FAC(50),  C(20),
     *                       DELTA(514),  WEIGHT(514)
      ERR=1.
      A(1)=1.0
      DO  1  I=1,NOP
         A(I+1)=FAC(I)*C(I+1)
         ERR=ERR+A(I+1)**2
1     CONTINUE
      FAC(49)=ERR
      DO  2  I=NOP+2,514
         A(I)=0.0
2     CONTINUE
      CALL  FFT(A,N)
      DO  4  I=1,NOS
         S=CDABS(DCMPLX(A(2*NOB(I)-1),A(2*NOB(I))))
         IF  (S-DELTA(I))  4,4,3
3           ERR=ERR+WEIGHT(I)*((S/DELTA(I))**2-1.0D+0)
4     CONTINUE
      FAC(50)=ERR-FAC(49)
      ERROR=ERR
      RETURN
      END
```

Figure 4.4.1.c : Minimum Power Gain Error Function.

While using the basic optimiser as described above, several additional features were tried:

1) To reduce the complexity of the problem, the first coefficient was not optimised but was set to unity, leaving the number of parameters to optimise equal to the order of the filter.

2) To satisfy psycho-acoustically weighted SNR measurements, the error function was modified to make comparison of the frequency response to an array representing user-specified frequency domain specifications (eg.: Modified E-weighting [VAN89] as read in from a parameter file).

3) To allow specific bands of the frequency response to be treated with individual priority, an array representing weighting as a function of frequency was incorporated.

4) To speed up calculation of the frequency response a radix 8 FFT was used in place of the radix 2 FFT, and the order of the FFT and the number of points evaluated from it were reduced to empirically found minimums.

5) To improve the scatter of the set of initial guesses generated from each start point, these were randomised, and to improve the settling of the routine to a final solution, the scale of the random scattering was reduced after each run of the optimiser (this will be referred to as a 'Freezing Simplex').

6) To avoid wasting time on solutions that are non minimum phase (and hence known to give higher power gain than necessary for the current shape), each solution can be converted to minimum phase before being evaluated (this option was only found to help if the initial start point was poor).

7) To simplify the optimisation problem by using orthogonal vertices in the simplex, the initial coefficients were converted to the zeros of the NTF (see programme 'laguerre.for' in appendix A3). Fortunately, this does not increase the complexity of the optimisation since the number of free parameters does not increase for real valued coefficients, however this option was only found to be useful in a few special cases. In general, the slight change of all the roots caused by changing one coefficient is better for avoiding premature termination of the optimisation in a local minimum.

The best features of these modifications are incorporated into three further programmes: 'mbnsopt.for', using (1,2,3 & 4), 'rsnsopt.for', using (1,2,3,4 & 5), and 'root_opt.for', using (1,2,3,4 & 7). The optimisation was found to inherently produce minimum phase solutions, so (6) was rejected to improve programme speed; conversion of the initial guess to minimum phase was found more efficient.

Variants of the main programme calling the optimiser were also written to permit keeping some coefficients as constants ('initnsopt.for'), to allow restrictions on certain coefficients (eg.: limited wordlength: 'insopt.for') and to optimise *performance* of a complete noise shaper with a user defined input signal (instead of NTF shape - 'signsopt.for'). These modifications were only found to be useful under certain conditions. Notably, the optimisation of the performance of a complete noise shaper is very slow (longer FFTs are required for accuracy in the error function) and is very easily stopped prematurely by a local minimum. This suggested that optimisation of the performance of a complete PWM DAC was out of the question since the time to evaluate a trial point would have to increase by 200 fold to accommodate decimation of the PWM output for sufficient accuracy in the error function.

The simplex optimiser is particularly good for low order filters and filters with simple shapes. Designing a noise shaper to complement the PWM unit which is expected to follow it requires a more detailed shape (using a higher order optimised filter). In such cases the simplex tends to stop short of finding good filters and the design takes a long time. This is not surprising since the simplex will stop in local minima unless it has built up some 'inertia' to straddle small surface effects. In high order systems with many conflicting (and non-linear) contributions to the error function, the group may simply not get moving. In these cases, global optimisation techniques should be used and this is looked at next.

4.4.2 : Simulated Annealing for Global Optimisation.

Three basic techniques for global optimisation were considered. Firstly an intensive search can be carried out, especially if integer coefficients of low wordlengths in a low order filter are required. This technique rapidly becomes prohibitively slow with high wordlengths or high order filters (>5) since the combinatorial possibilities are enormous. An intensive search can be used in the region of a

good solution but this is not guaranteed to be global and of little use in this application.

The second technique is that of the genetic algorithm (GA). While this guarantees good solutions, and can be left to continue improving even after a 'good' solution has been found, the algorithm is best suited to limited-wordlength implementations. GA is slow to converge and if encouraged to proceed more quickly by starting with filters closely approximating the desired response, tends to skip the best solution because the filter error surface has severe discontinuities with each 'good' solution having distinct differences from the best.

The third technique is that of simulated annealing (SA). This technique can be proved to asymptotically approach the best solution as the amount of time available is increased, and has been shown to be useful for IIR filter design [MAR92]. This was chosen as the most promising.

The fundamental operation of SA is well known, relying on random moves from the current state being accepted if they offer better solutions to a problem or are acceptably worse. The acceptance criterion is commonly the Metropolis Criterion [MET53] which is gradually made tighter, analogous to gradual temperature reduction in an annealing solid. With this criterion it is claimed that the final solutions will be globally optimal if it is not tightened too quickly. This optimiser was programmed as a replacement for the simplex in the programme 'nsopt' but soon showed that the technique of simulated annealing is *very* computationally expensive and only becomes useful as a last resort. Fortran code for subroutines 'Anneal' and 'Metropolis' is shown below:

```
      SUBROUTINE ANNEAL(NOP,FAC,T,RT)
      IMPLICIT NONE
      EXTERNAL ERROR,FFT,R8TR,ORD1,ORD2,RAN2
      INTEGER NOP,I,N,IDUM
      DOUBLE PRECISION FAC(50),ERR,OLDERR,ERROR,RAN2,ERRLIM,T,RT,TMIN
      N=300
      TMIN=1.0E-8              ! !> 1.0e-4
      ERRLIM=1.0E-9            ! accuracy : O[6-9]
      IDUM=-6                  ! initialise RAN2
      ERR=RAN2(IDUM)
      ERR=ERROR(FAC)           ! create an initial error value
    1 CONTINUE                 ! end of set-up ... annealing follows
      OLDERR=ERR
      DO 2 I=1,N               ! loop with constant temperature, N times
         CALL METROPOLIS(NOP,FAC,T,ERR)
    2 CONTINUE
      IF (DABS(ERR-OLDERR).LT.ERRLIM) RETURN ! check it's still moving
      T=RT*T                   ! reduce the temperature
      IF (T-TMIN) 3,1,1        ! check it's not absolute zero
    3 RETURN
      END

      SUBROUTINE METROPOLIS(NOP,FAC,T,ERR,IDUM)
      IMPLICIT NONE
      EXTERNAL ERROR,FFT,R8TR,ORD1,ORD2,RAN2
      INTEGER NOP,I,IDUM
      DOUBLE PRECISION FAC(50),ERR,NEWFAC(50),NEWERR,T,DELTE,ERROR,RAN2
      DO 1 I=1,NOP
C each factor in turn is randomly modified by up to +/- step size:'T'
         NEWFAC(I)=FAC(I)+2.*T*(0.5-RAN2(IDUM))          ! perturb
         NEWERR=ERROR(NEWFAC)                            ! eval. error
C test for improved guess
         IF (NEWERR.LT.ERR) THEN                         ! ** ACCEPT **
            ERR=NEWERR
            FAC(I)=NEWFAC(I)                             ! new coeff
         ELSE
C test for possible usefulness (Metropolis criterion)
            DELTE=NEWERR-ERR
            IF (EXP(-DELTE/T).GT.RAN2(IDUM)) THEN        ! ** ALLOW **
               ERR=NEWERR
               FAC(I)=NEWFAC(I)                          ! new coeff
            ENDIF
         ENDIF                 ! ie: otherwise perturbation NOT accepted
    1 CONTINUE
      RETURN
      END
```

Figure 4.4.2.a : Subroutines for Simulated Annealing Optimisation.

To overcome the slow progress made by a simple simulated annealer, four modifications were made to its basic operation. A flow chart of these revised operations is shown in figure 4.4.2.c .

1) N, new random moves are made in each ordinate in turn at one given 'temperature' before any change can be made to the temperature.

2) The scale in each ordinate of the random moves is calculated according the proportion of the successful moves in the last set of N random moves. This proportion is applied to a scale modification function so that the scale increases by up to 3x for total acceptance or decreases to 1/3 for total failure. This ensures that the optimiser adapts to the surface it is following [COR87], (see below - 4.4.2.b).

3) The scale is modified R times before the temperature is reduced, where R is varied linearly in a limited range. This range is a function of the standard deviation of the accepted points' errors, while searching on the last temperature plateau [WHI84]. This standard deviation can be used to check the progress of the annealer (see figure 4.4.2.d for a typical plot).

4) The algorithm is restarted from the best found point after K temperature reductions. If the optimiser jumps away from a good point just before the temperature is reduced (and then doesn't have the energy to jump back), the best point is not lost [BEN92].

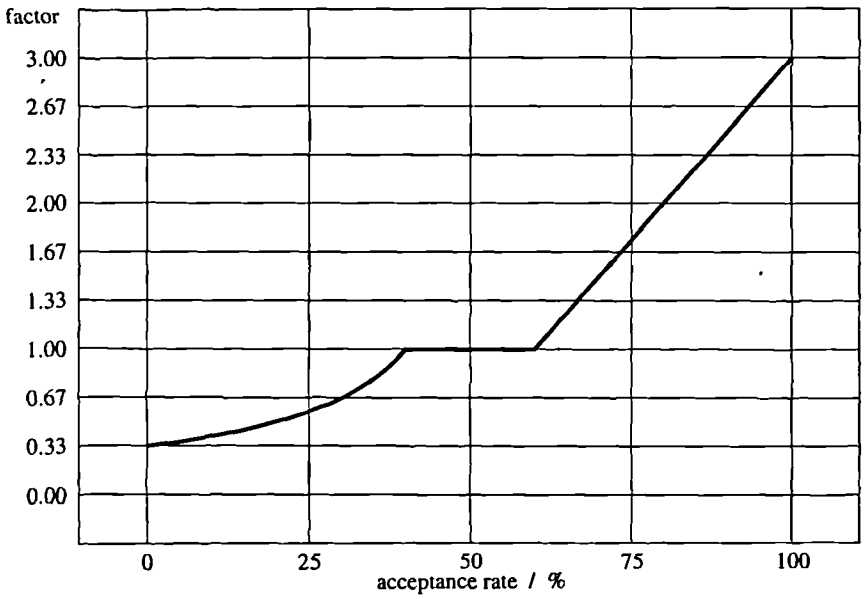


Figure 4.4.2.b : Scale Modification Function.

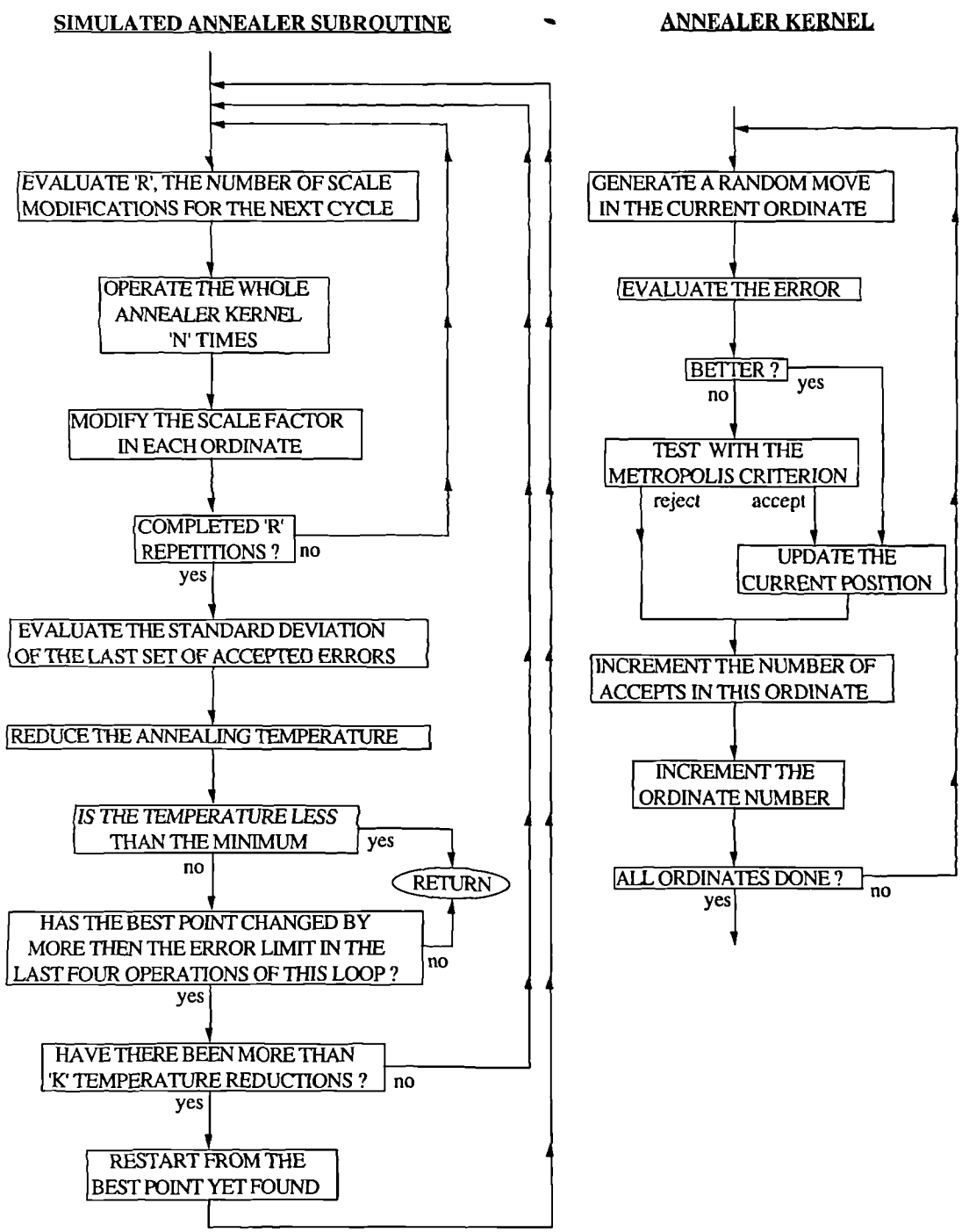


Figure 4.4.2.c : Modified Simulated Annealer Flow Diagram.

With these modifications the annealer proceeds some 8-10 times faster (actual CPU time) than the basic annealer and produced solutions identical to within the error limit (1.0×10^{-6}). This programme, 'sansopt.for', becomes useful for orders above 10 where the simplex begins to fail (and has to be repeated a lot), and almost essential above 15 where the simplex can almost be guaranteed to stop short of a good solution. The modified annealer is still extremely slow, taking 2 hrs., 40 mins. for a fifth order filter and 10 hrs., 6 mins. for an eleventh order filter (on a Sun-Sparc, IPX).

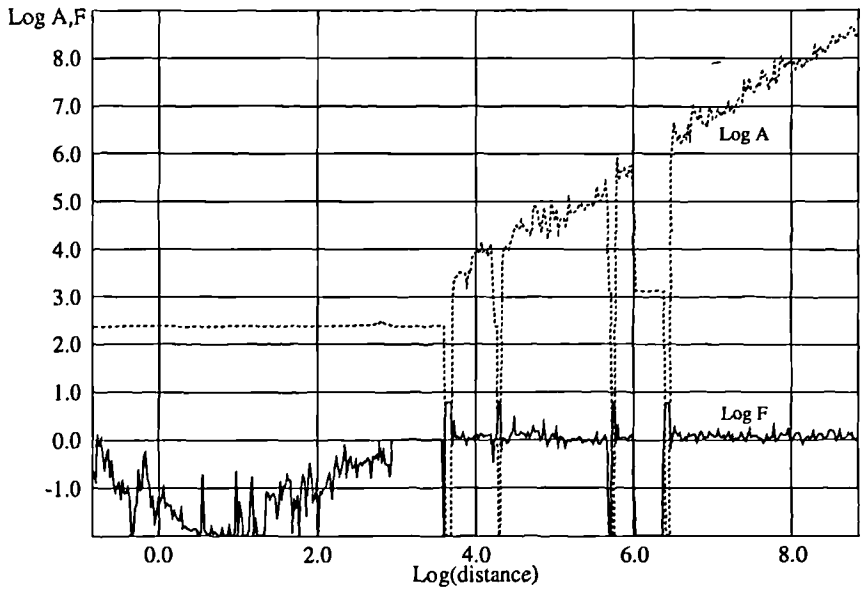


Figure 4.4.2.d : Plot of Annealer Progress through a Fifth Order Optimisation.
(Trial solutions' error average (F) and standard deviation (A) vs. 'distance' from previous solution).

4.4.3 : Pattern Search Optimisation for Limited-Wordlength-Coefficient NTFs.

With the experience that the simplex optimiser was only effective for low order systems (<20) and the simulated annealer was extremely slow, a new optimisation routine was sought to assist in the conversion of floating-point designs into limited wordlength implementations. The pattern search technique [HOO61] was adopted for its rapid descent to good solutions and its ability operate on quantised surfaces (ie. error surfaces patterned by quantisation noise).

The basic principle of the pattern search is to attempt moves from a start point looking in a random direction (by modifying the ordinates in turn by a step size). If any ordinate change yields a better solution than the 'base' then that change is accepted. After attempting modification of each of the ordinates, the difference vector from the initial 'base' to the new best point is calculated and called a 'pattern'. This vector is then repeated from the best point, and modifications are started around the new point. Should the new point fail to have any better points around it, then the pattern is dropped, and more points around the best point are sought. If successful, the pattern can be extended from there. If unsuccessful the step size is reduced until the moves can continue or the step has become acceptably small.

One modification was made to this optimiser to improve its convergence. To avoid the first coefficient always being the first modification which is tried, the order of the ordinate modifications was shuffled each time a new point was found. This prevents premature termination of the search arising from sub-optimal modifications to the first coefficient being accepted as a result of a subsequent coefficients being changed. Since the search may become stuck in local minima, the optimiser is re-applied, often up to 100 times, using the best point found as a new initial guess. The whole programme is listed in appendix A.3.5 in FORTRAN, and called 'hjfnsopt.for'. A flow diagram for the optimiser is given below:

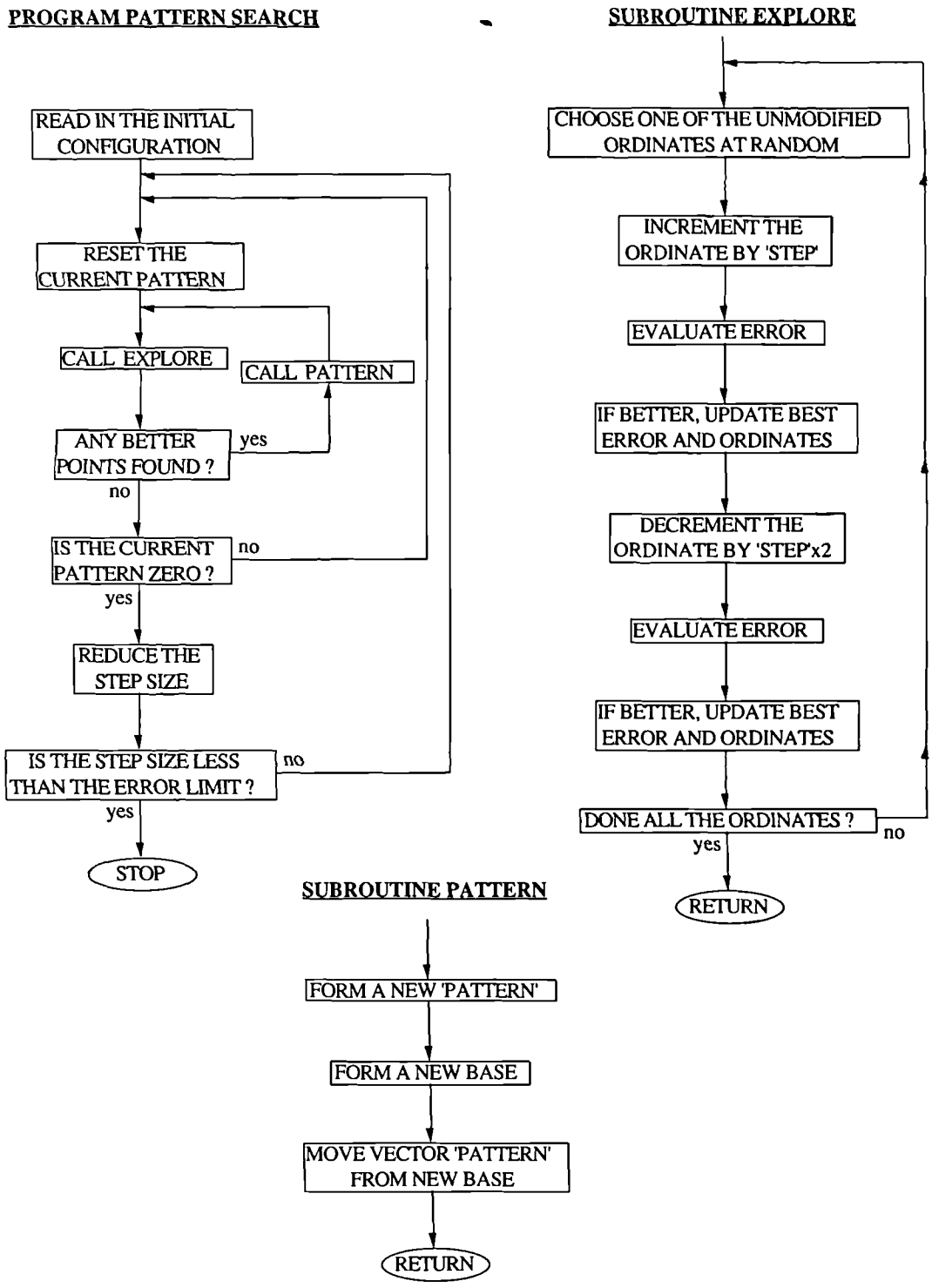


Figure 4.4.3.a : A Flow Diagram of the Operation of the Modified Pattern Search

A simple modification to the error function of this programme allows it to be used to find limited wordlength solutions to approximate the optimal floating-point NTF. Since the coefficients of the NTF tend to be of similar magnitude to their neighbours (ie. progressively reducing size with index), the direct form structure of the feedback filter was modified to allow greater coefficient accuracy:

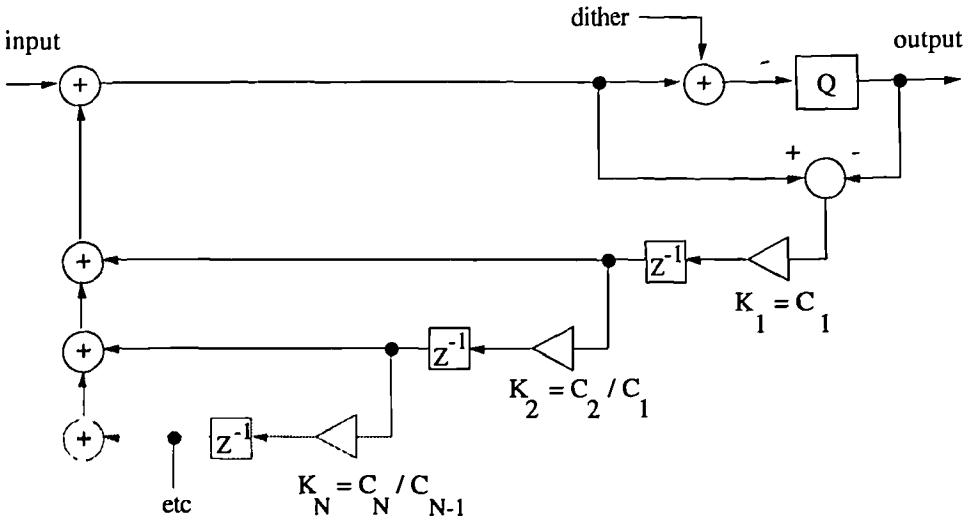


Figure 4.4.3.b : Modified Noise Shaper : 'Cascade' Feedback Filter Structure

This structure was written into the truncation code added to 'hjfnsopt.for' and the new programme, 'hjinsopt.for' can be used to optimise for limited wordlength coefficients.

4.4.4 : Characteristics of Optimised NTFs.

Obviously minimum power gain (for given stopband requirements and order) is one of the characteristics that the optimised filters exhibit, but some other interesting features became apparent by analysing the output of the optimisers. Complicated frequency response requirements sometimes produced unusual results, but for a single stopband high pass filter, particular characteristics can be recognised in the frequency response or organisation of zeros on the z-plane.

The filters always exhibited minimum phase except where the optimiser had got stuck in a local minimum. Where a poor initial guess and a small number of iterations were used this might not occur, and in fact minimum phase became useful as a quick check on the suitability of an initial guess. This result is not surprising, considering the link already known between these two (see section 4.3), but it does serve to confirm that the optimisation routines have been operating correctly.

Another notable recurrence was the presence of a zero at Fs/2 in all odd order filters, and a tendency for zeros in low order filters to be placed near the edge of the stopband. This is consistent with maintaining a quick transition band which is a key to making area 'A' (see figure 4.3.2.b) more rectangular and hence smaller. Low order filters also tend to use zeros entirely in the stopband, organised in an arc of a circle of radius greater than one and centred on the real axis at greater than two.

One interesting observation was that in the higher order filters (where there are 'spare' zeros to tightly control the stopband ripple) the passband was not flat but mildly rippled, while selective zeros (large radius) were placed at the transition band edge. This would suggest that there is more to be gained by a rapid transition band than can be saved by a maximally flat stopband. In higher order filters still, non-resonant zeros would appear, almost equi-spaced in frequency across the passband. Typical frequency response and z-plane graph for a fifth order simplex-optimised filter are shown below as an example:

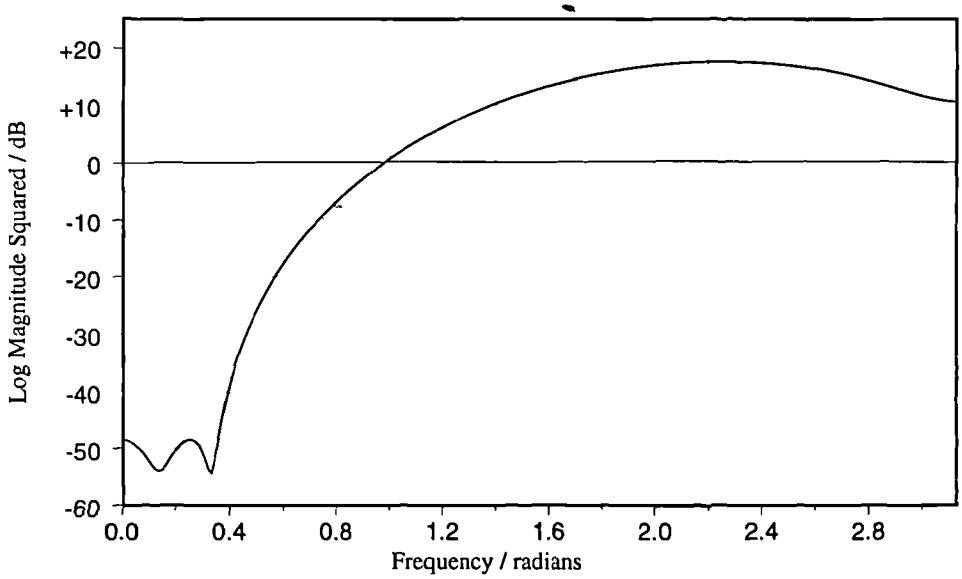


Figure 4.4.4.a : Frequency Response of a Fifth order Optimised NTF

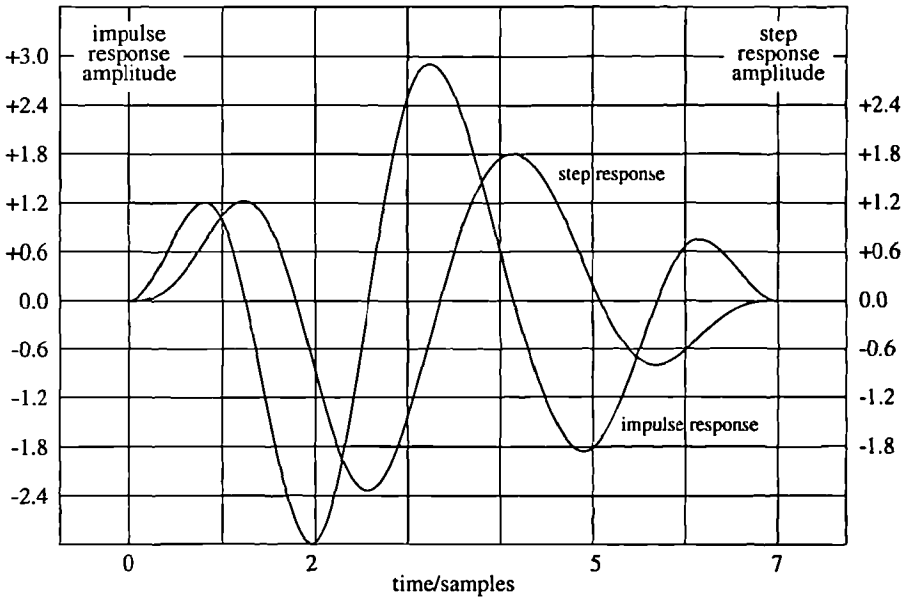


Figure 4.4.4.b : Impulse & Step Responses of a Fifth order Optimised NTF

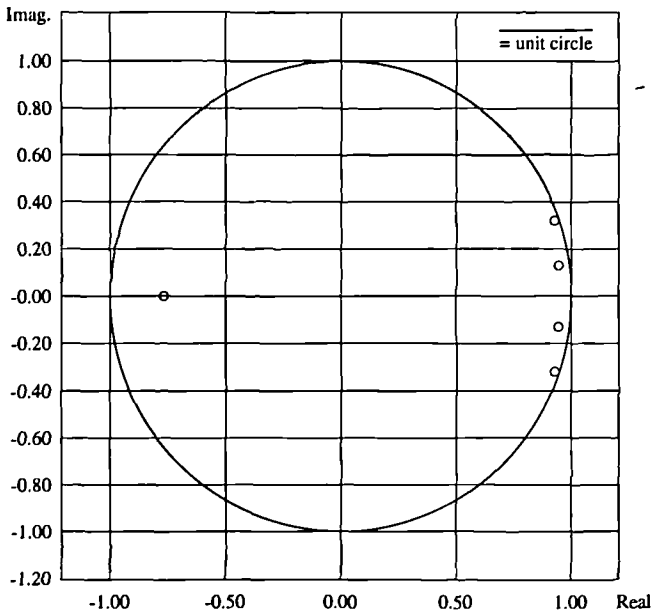


Figure 4.4.4.c ; Organisation of Zeros On the Complex-Z Plane for a Fifth order Optimised NTF

4.5 : Noise Shapers With Special Properties.

Some DAC applications can benefit from noise shapers with noise transfer functions that have special properties, either in the time domain or frequency domain, or both. Wherever noise shaping is used, the additional noise added by the shaper can upset the operation of subsequent stages of a DAC leading to a reduced overall performance. For PWM DACs this is particularly true when a high noise power level or a well correlated noise (sample to sample) is produced. Special time domain or frequency domain features in the NTF can reduce these effects, and in some cases novel architectures can help further. Seven particular cases of interest will be looked at in this section :

1) NTFs for use as feedforward filters (cf. section 4.3.4),
2) NTFs for use in closed loop PWM DACs (this is studied further in section 5.4.2),
3) NTFs with IIR implementation,
4) NTFs with reduced complexity by forcing some coefficients to zero,
5) NTFs for use with two sample PWM types (DSPWM, 2SCPWM),
6) NTFs to avoid noise-carrier intermodulation, and
7) NTFs to avoid signal-noise intermodulation.

4.5.1 : Mixed Domain Optimisation : 'Predictive' Noise Shaping.

As shown in section 4.3.4, very low gain noise shaping can be achieved if the error introduced by the quantiser in a noise shaper can be spread in advance of the sample from which it was derived as well as spread on subsequent samples. To avoid destroying the low resolution achieved by the quantiser, errors added to advance samples must either be taken into account iteratively or must be integer values within the output wordlength. By looking at the difference between the quantiser input, and the output, the amount of error signal added to the output stream can be assessed. If this error is large, a lower value can replace it by re-spreading the error across more samples. A filter to spread the error in this way, can only take integer values off the feedback signal without changing the current error. Thus, both the feedforward coefficients, and the coefficient aligned with the current sample must be integers. For example, using a 16 bit input / 8 bit output noise shaper with a truncating quantiser and hexadecimal notation, if

the current feedback value is :	H(z).E(z)	=	0286	Equation 4.5.1.a
the current input value is :	I(z)	=	4292	Equation 4.5.1.b
the input to the quantiser is :	I(z)+H(z).E(z)	=	4518	Equation 4.5.1.c
the output from the quantiser is :	O(z)	=	4500	Equation 4.5.1.d
the quantised input is :	MSByte(I(z))	=	4200	Equation 4.5.1.e
thus the quantised excess is :	O(z)-MSByte(I(z))	=	0300	Equation 4.5.1.f

In this example, the excess (0300H) could be spread, not just on subsequent samples (as in a purely recursive noise shaper) but also in advance of the current sample in units of 0100. To find out the best way of spreading this excess, filters with integer values for coefficients $C_{-k}...C_0$ have to be generated, where k is the number of feedforward stages in use. These filters can have floating point coefficients for C_1 ... C_{n-k}, since filtered error added to subsequent samples does not have to be quantised. If only one

advance coefficient (ie. k=1), and a fourth order filter are used (ie. n=4), it might look like :

$F(z) = \{ C_{-1} = 1, C_0 = -2, C_1 = 6.5\text{x}10^{-1}, C_2 = 2.5\text{x}10^{-1}, C_3 = -9.3\text{x}10^{-2} \}$ Equation 4.5.1.g

In this case, when coefficient C_0 is applied, 0200 can be subtracted from the output, so that the excess is reduced to 0100. If this is done, C_{-1} should be added to the previous output value (ie. add 0100) and the rest of the filter F(z) will contribute to the next feedback error values is the same way as H(z) is usually operated.

By looking at the impulse response of high order versions of H(z) for purely recursive noise shapers, the ratio between the first coefficient and subsequent ones can be found which most suits the frequency response required from then filter (in fact, C_0 =1, so the ratios equal the coefficient values). From this the rounded coefficients can be taken as the closest estimates of integer coefficients that could be used in a filter where k>0. By fixing k+1 coefficients at their rounded values, optimisation of the remaining coefficients can be restarted to produce a feedforward filter using the same frequency specifications as the H(z) from which it is derived suitable for use as F(z).

For example a noise shaper operating at 8x oversampling reducing the input wordlength of 16 bits to an output wordlength of 8 bits requires a stop band approximately 47.5 dB lower than the 8 bit (output) quantisation floor (as calculated using equation 4.3.3.i). If filters of various orders are optimised exhaustively to the same solution, their gains vary as shown below :

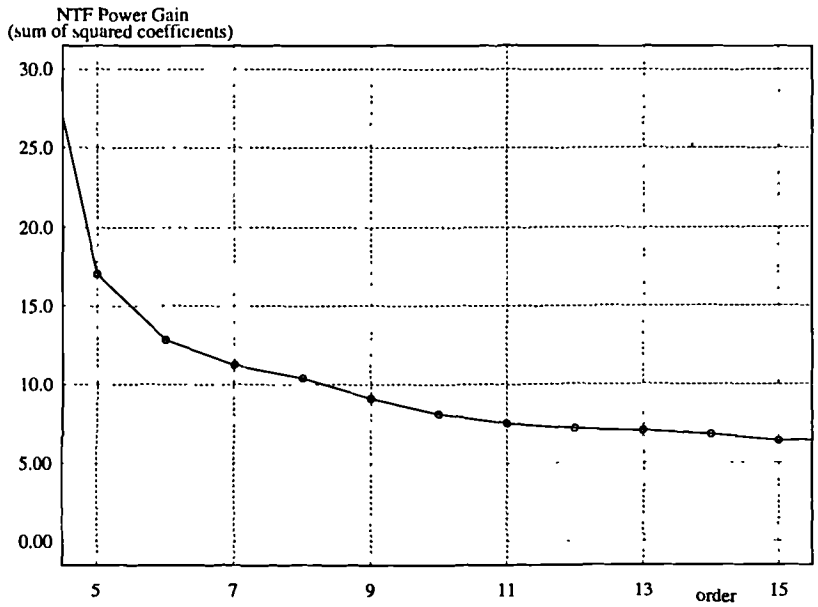


Figure 4.5.1.a : Plot of Power Gain vs. Order For a Family of Purely Recursive NTFs
(nb.: the power gain approaches a limiting value...higher order filters offer little improvement)

From the above set of noise shapers, the rounded coefficients can be chosen for designing a feedforward filter with , say, K=1; the plot of second coefficient value vs. order shown below suggests the first two coefficients should be : $C_{-1} = 1$, $C_0 = -3$ for filters with order circa 5, or $C_{-1} = 1$, $C_0 = -2$ for filters with order ≥ 15 :

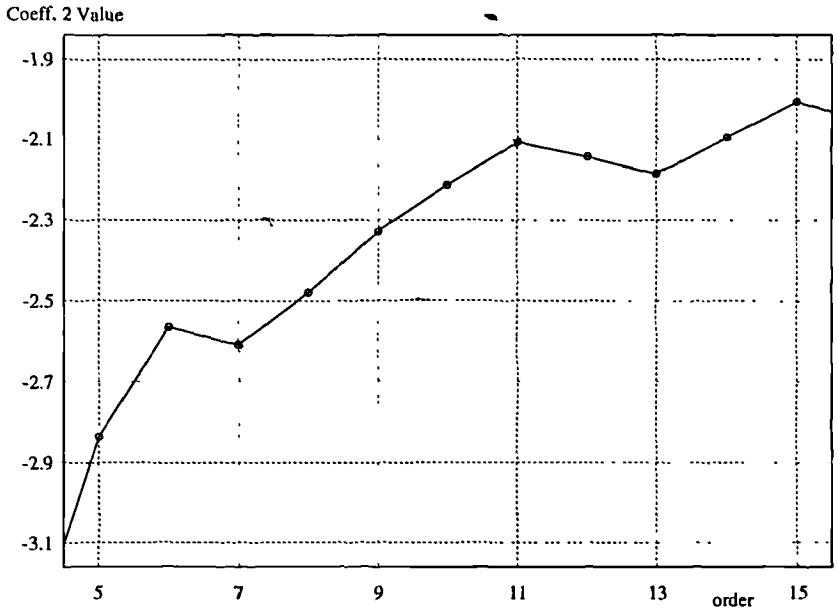


Figure 4.5.1.b : Plot of 2nd. Coefficient Value For a Family of Purely Recursive NTFs
(nb.: all the higher order filters have very similar initial coefficients)

Taking these values (1,-3 and 1,-2) as the first two coefficients, new sets of filters can be found by optimisation with power gains as shown below :

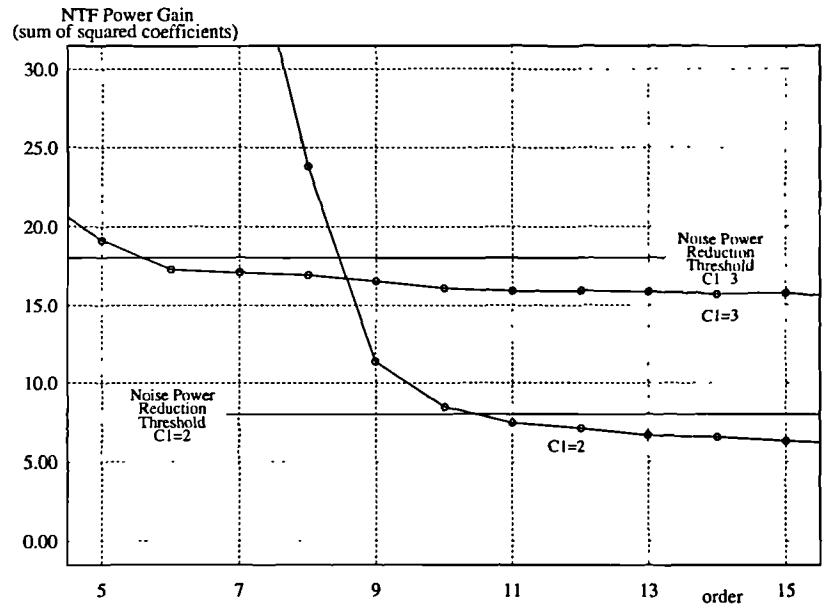


Figure 4.5.1.c : Plot of Power Gain vs. Order For Two Families of Feedforward NTFs with K=1
(nb.: the power gain still approaches a limiting value, but is *higher* than before)

The purely recursive filter and the recursive - feedforward combination can be compared by simulation; as expected, a noise power reduction has been achieved (since the error can now be spread both before and after the sample from which it originates). With both 'sides' of the sample being used to disperse the error, the error can be placed more closely to the sample from which it was quantised, *ie.* the total error delay has been minimised. The spectral improvement is shown below :

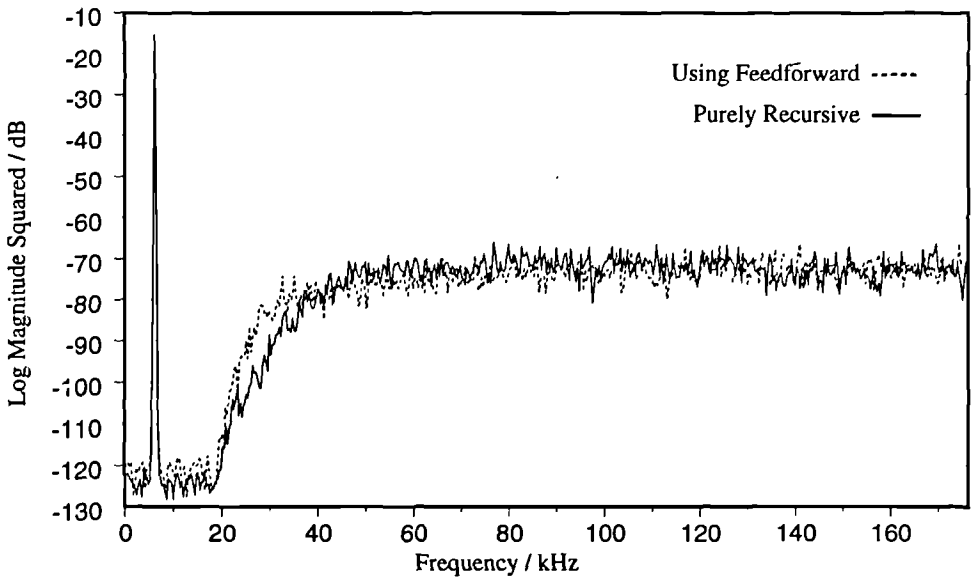


Figure 4.5.1.d : Noise Shaper Output Spectra for a Purely Recursive and a Feedforward NTF

4.5.2 : Mixed Domain Optimisation : 'Deferred Evaluation' Noise Shaping.

Closed loop PWM DACs will be studied in section 5.4. These can operate by placing a PWM directly after the quantiser in a noise shaper so that the effects of the PWM are corrected for as well as the effects of the quantiser. Since the PWM increases the sampling rate at the output, decimation of the output is required to permit subtraction of the output from the input to find the error feedback signal. This decimation adds delay to the feedback loop which modifies the NTF. Forcing some of its initial coefficients to zero can correct for this. Coefficients with zero value need not be evaluated, and if omitted, allow for a time shift in the remaining part of H(z) as shown below:

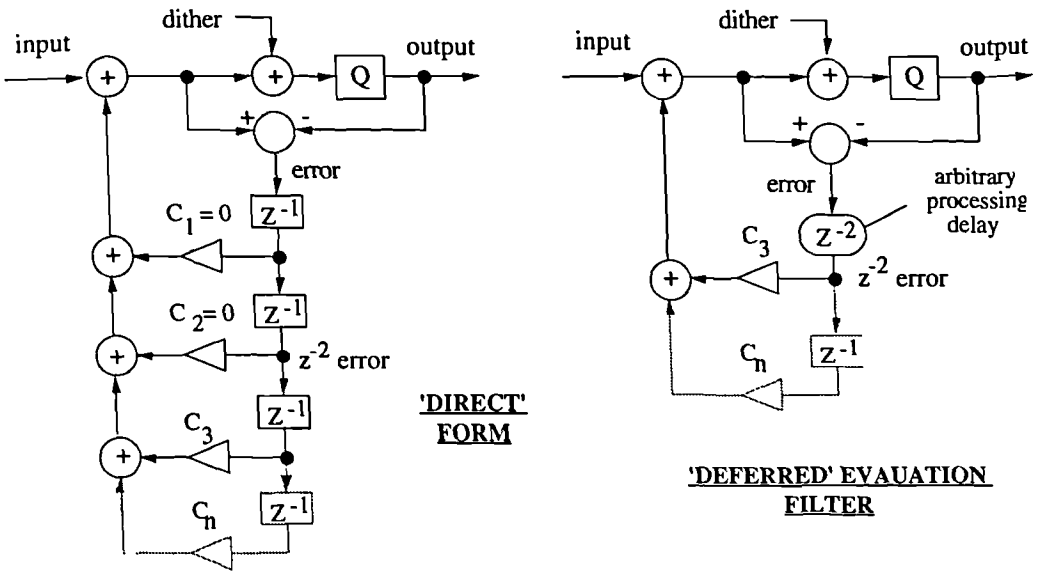


Figure 4.5.2.a : Direct Form and 'Deferred' Noise Shapers

The leading '1' in the NTF is not present in H(z) (cf. equation 4.1.2.d), so this coefficient cannot be altered (it is inherent to the structure). This leaves the higher indexed coefficients to work with, although it is the first few coefficients which are most useful. Nulling the first, or first few, coefficients of H(z) permits a 'deferral' of calculating the feedback value which can accommodate the additional loop delay in closed loop structures. A noise shaper, using such a filter will be referred to as a 'Deferred' noise shaper.

As discussed in section 4.3.4, small noise power-gain is achieved with low delay filters, so adding delay in the feedback loop will cause the NTF to have a higher power gain than usual, making it even more important to optimise the NTF for minimum power gain. Without optimisation, the design of deferred NTFs becomes very difficult, and the problems of overload in the quantiser and restricted input range become major obstacles to using of these structures.

Nulling the first few coefficients can be done very easily in optimisation, since the subroutines have been written to work with multiplicative modification of the coefficients rather than the coefficients themselves. Initial coefficients sets such as $\{1,0,C_2,C_3..C_n\}$ or $\{1,0,0,C_3,C_4..C_n\}$ will result in unnecessary attempts to multiply the zero-valued coefficients by a modification factor but the coefficients will remain at zero as required. Slight reduction in the speed of design may be suffered since the zero-valued coefficients are still used in calculations for ordinate modification. The zero-valued coefficients represent a reduction in the number of degrees of freedom in the filter so a higher order filter is often required. Furthermore, the increased power gain of such filters (due to increased delay) may be unacceptable unless high oversampling ratios can be used. If the required design has tight constraints on wordlength and oversampling ratio, the filter order can sometimes be increased to bring the power gain back to a reasonable level.

Filters already optimised for non-deferred designs make bad starting points for the optimisation of deferred filters. Taking the mid section of a linear phase filter of the required frequency response and twice the order, has been found to provide starting points that quickly lead to acceptable solutions. The filter has to be appropriately prepared which involves truncating the length to the required order, scaling to make the new first coefficient unity, and zeroing the required coefficients (eg. C_1). Since the impulse response in the deferred NTF can only be guaranteed zero at the zero-valued sampling instants, processing which does not have an integer number of samples' group delay will destroy the operation of the NTF. Thus, the decimation filter group delay must be constant over the signal band (ie., a linear phase operator) and integeric at the sample rate of the noise shaper (ie. not an arbitrary design of multi-rate processing elements). This aspect of closed loop PWM DAC design will be examined in detail in section 5.4 .

4.5.3 : Mixed Domain Optimisation : IIR NTF Design.

Up to this point the feedback filter used in noise shapers has been designed and demonstrated using finite impulse response (FIR) examples. With the additional flexibility brought by optimising filter coefficients (rather than using sinusoidal deigns) better filter architectures can be employed. In general, recursive filters can achieve filtering with faster transition bands than non-recursive filters for the same computational effort as a result of the extra freedom introduced by using a transfer function

with poles as well as zeros. However, this additional capability comes at a price : the filter itself can become unstable if inappropriately designed, and will tend towards idling patterns with periodic characteristics. To evaluate the usefulness of such a filter, a modified direct form II structure was chosen (see chapter 3) and coded into a revised error function for use with the optimisers discussed in section 4.4.

The error function for optimising such a filter can still be based on a weighted balance between power gain and stopband failure but further considerations have to be included to prevent unstable filters being produced. Assessment of the power gain of the filter becomes less efficient than in the FIR case since the impulse response may be long. An estimate of the power gain can be found by two methods using either the discrete time domain or the discrete frequency domain. In the discrete time domain, the impulse response series has to be truncated, in which case the accuracy of the answer has to be traded for the computational complexity of its evaluation. In the discrete frequency domain, after the numerator and denominator of the transfer function have been transformed separately, the frequency response can be found by division of each frequency bin in turn and then squared and accumulated. In this case the accuracy is limited by time aliasing which will occur in the discrete transform of the numerator. Although the frequency domain technique is slightly more efficient in non-resonant filters (or for the same effort is slightly more accurate), it can produce gross errors where evaluation of the discrete Fourier transform of the denominator is near a pole in the transfer function. Since poles naturally migrate during the progress of an optimisation, this condition has to be checked for and such additional overhead makes this technique less desirable generally. For this reason, time domain evaluation is preferred; an example error function coded in Fortran in listed in appendix A.4.4.

The recurrent characteristic observed in the results of optimisation was that a resonant pole-zero pair at the edge of the noise passband permits the use of less resonant poles and zeros elsewhere in the response. Thus most of the poles and zeros are less resonant in the IIR designs than the zeros is similar FIR designs; this commonly leads to slightly reduced noise power gain. An example of this is shown below :

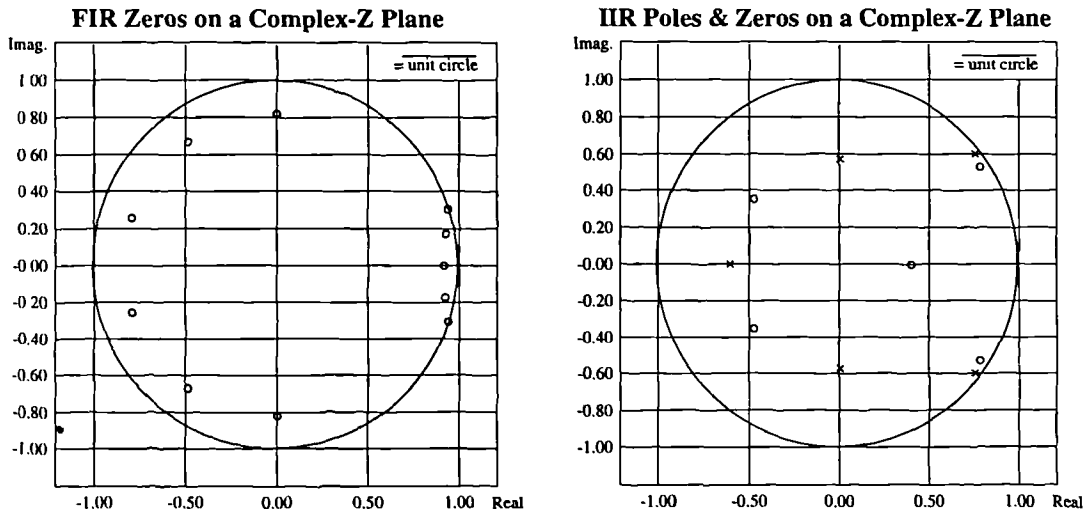


Figure 4.5.3.a ; Organisation of Poles and Zeros in Similar FIR and IIR NTFs.

The frequency responses of these two filters is shown below (figure 4.5.3.b), and here the dominant resonance in the IIR filter can be seen near 40 kHz. Although this resonance has effectively widened the noise passband (by speeding up the transition band) it has not significantly improved the power gain of the filter when compared to the FIR case. This is probably because the dip in the frequency response near 50 kHz sacrifices the gains achieved by the faster transition band. Since a zero is always required at a frequency just above the dominant resonance, to flatten the noise passband, the benefits of IIR NTFs are small unless high oversampling is used (ie where fast transition band becomes more important). This conflicts with the requirements of a digital amplifier where large oversampling is undesirable, so IIR NTFs are not as favourable in this application as would otherwise have been thought.

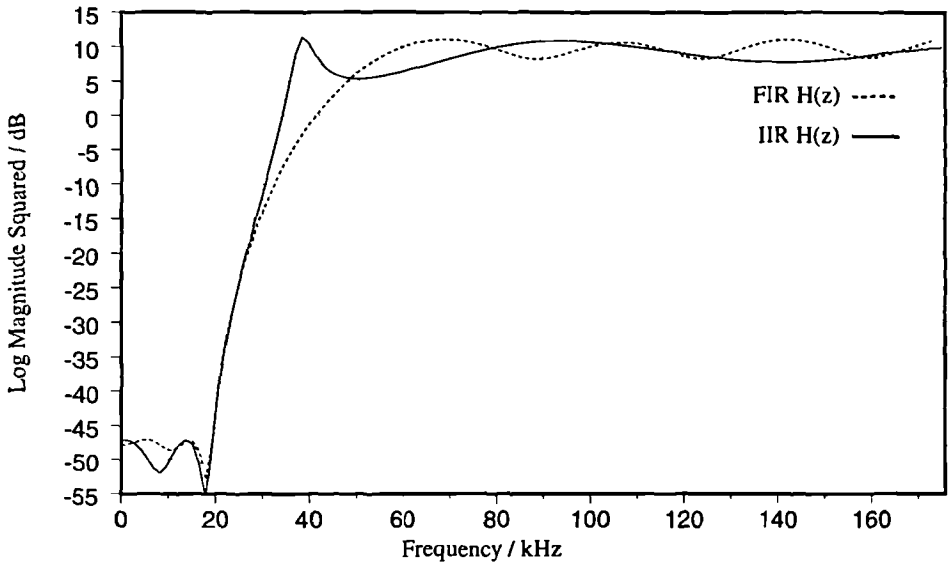


Figure 4.5.3.b : Frequency Responses of Similar FIR and IIR NTFs.

The dominant resonance in the IIR case commonly leads to high noise power gain near the signal band. Unfortunately, PWM is not a completely linear process and intermodulation can occur between this noise and the signals in the audio band, so high noise power gain near the signal band is particularly undesirable, making IIR NTFs no better than FIR NTFs for low-oversampled systems. Minor noise power gain reductions can be achieved for higher order filters, leading to fractional improvements in SNR that can be achieved after a PWM for signals which are small enough not to trigger intermodulation with the noise. A plot demonstrating the trend of power gain vs. order is shown below (figure 4.5.3.c), superimposed on the FIR summary shown in figure 4.5.1.a, but re-expressed in terms of multiplies per sample instead or order. This permits fair comparison of the computational complexity of the two systems, since additional multiplication is required for evaluation of feedback, as well as feedforward, coefficients in the direct form II architecture. An odd number of multiplies is always required for the IIR filters since there is a loop gain and 2*N multiplies used (where N is the number of delays in the filter). Different architectures may be capable of slightly better performance and different filter requirements lead to slightly different comparison graphs but the overall trend shown here has been found to apply for a very large number of cases.

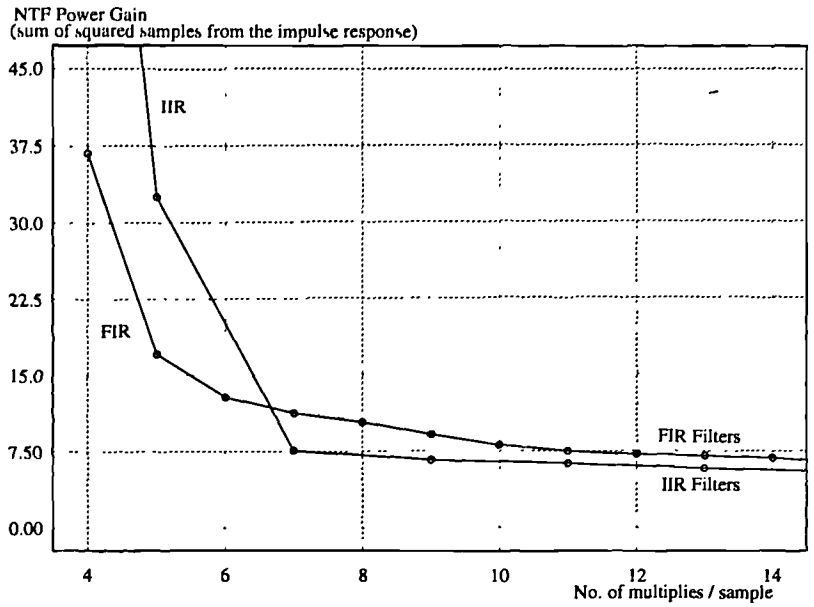


Figure 4.5.3.c : A Graph Comparing Filter Complexity for a Family of FIR and IIR NTFs.

4.5.4 : Mixed Domain Optimisation : Reduced Complexity NTF Design.

Optimisation of filter coefficients permits the particular value of some coefficients to be fixed while other coefficients are modified to accommodate such a restriction as has already been seen for predictive and deferred evaluation NTFs. Setting a coefficient to zero is of particular interest since this permits a reduction in the filter complexity.

In many cases of FIR optimisation the sampled impulse response of the resulting filter will have near zero values for some coefficients, and by setting these value to zero and proceeding with further optimisation, filters of similar performance and lower complexity can be found. Shown below is the impulse response of a ninth order FIR NTF before and after such treatment (the stopband gain and power gain are almost unaltered). In this case, a large coefficient has been zeroed to demonstrate the effect clearly (C_6), but in most cases smaller coefficients are chosen (eg. C_7).

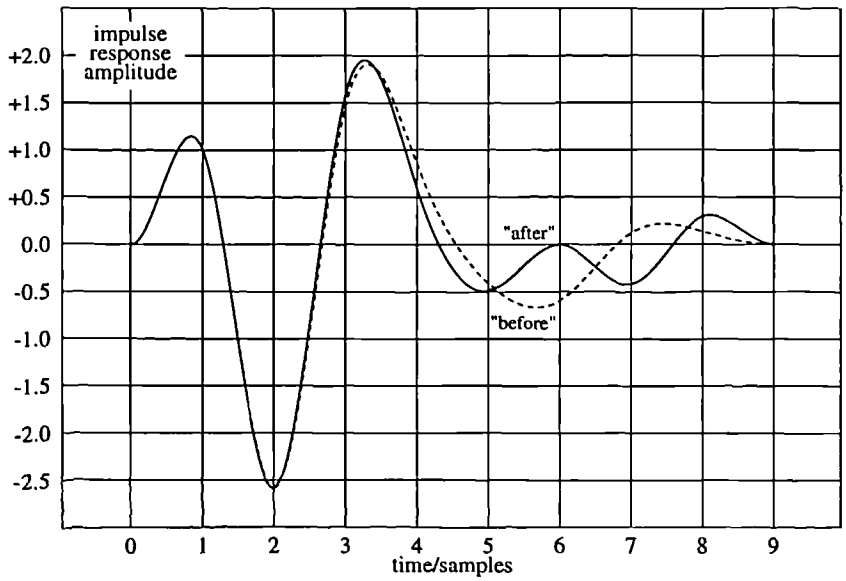


Figure 4.5.4.a : Impulse Responses of an FIR NTF Before and After Coefficient Zeroing.

In the IIR case, it is less clear which coefficients can be eliminated but commonly, those which are found to be near zero (<0.1) can be set to zero and accommodated in a re-design. A summary graph is presented below (figure 4.5.4.b) showing gains that have been achieved for the family of NTFs presented in figure 4.5.3.c. For the special cases where coefficients have been zeroed the power gain is usually slightly larger than that for the filter from which it was derived, so each filter (represented by a dot) will appear horizontally, one or two multiplies to the left of its 'parent'. Reduction of complexity by three multiplies has not been included but for such low orders this usually results in substantially increased power gain.

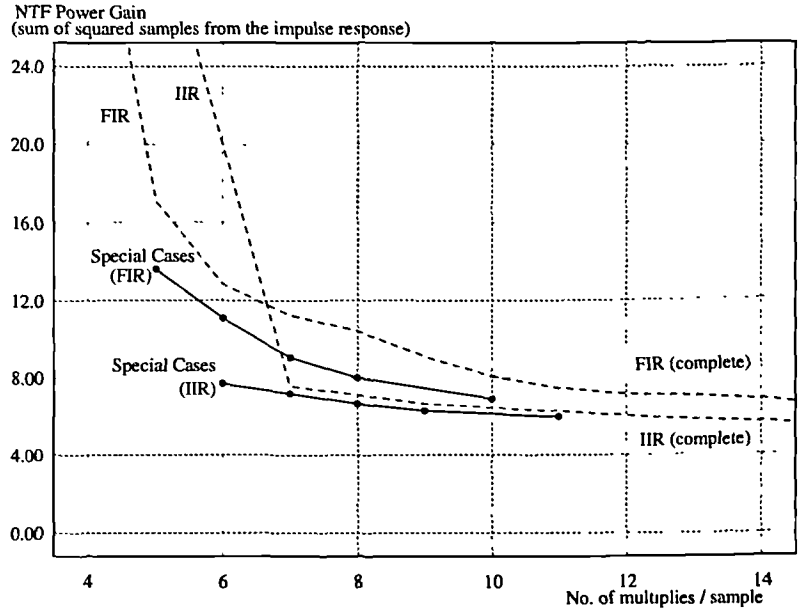


Figure 4.5.4.b : A Graph Comparing Filter Complexities for Special Filters with Zeroed Coefficients

4.5.5 : Zero Interleaved NTFs.

PWM DACs which use two sample modulation generally have a signal *vector* comprising the amplitude and direction in time (edge) information. Noise shaping assumes the input signal to be scalar, ie. the noise shaper does not take into account which edge the output may be supplied to. This inability to handle the signal as a vector quantity can lead to cancellation between energy truncated from one sample and the energy of the sample to which the (filtered) error is added.

Taking 2SCPWM and a first order sinusoidal NTF as an example, $H(z) = z^{-1}$, the errors arising from treating the signal as scalar become obvious. Assuming an initial error value of zero, but an input which does not exactly lie on a quantisation level of the output, there will be a non-zero error value to add to the second input. If the first sample is used to modulate, say, a leading pulse edge, then when the error from it is added to the second sample it will modulate the trailing edge (unless the second sample exactly lay on an output quantisation level). This means that except for special cases, error truncated from the leading edge (making it later in time), will be added to the trailing edge (making it later in time too) effectively shifting the whole pulse. Similarly, truncation of a trailing edge (making it earlier in time) may be added to the next leading edge (making it earlier in time too) effectively shifting the space between pulses.

To summarise, some of the noise shaped portion of the input signal will phase modulate the entire remaining signal. The noise shaping and two sided PWM combine to produce a phase modulator based on the amplitude quantisation; this will be referred to as a 'parasitic' amplitude-modulation-to-phase-modulation conversion process (parasitic AM->PM for short).

Constructing a noise shaper to take signal vectors into account is difficult, but since the PWM case only uses two basis vectors this can be done reasonably easily by using two noise shapers, one to shape the signal for leading edges, one to shape the signal for trailing edges. Using two noise shapers in this way at least doubles the noise shaper complexity, and requires NTFs in each shaper to be good enough to operate at half the original oversampling ratio (since each handles only alternate samples). Alternatively, sample rate increase can be provided to increase the sample rate by a factor of two before entering the dual noise shaper (although this removes the advantages of the lower pulse repetition rate inherent to two sample per pulse modulation types). The structure for such a system is shown below:

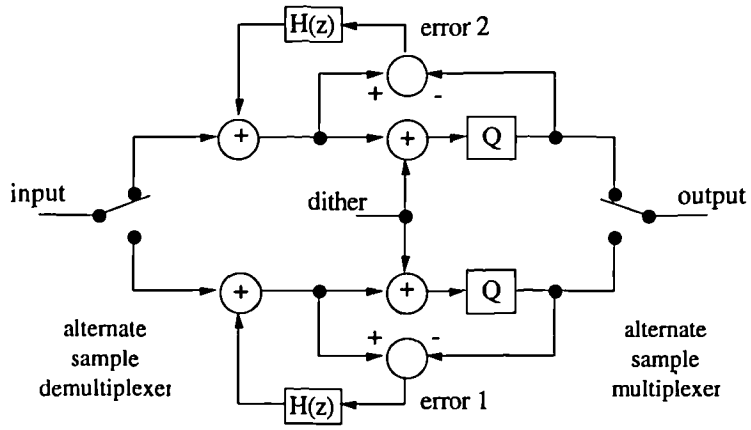


Figure 4.5.5.a : Two Noise Shaper Solution to Parasitic AM->PM Conversion in Two Sided PWM.

A more elegant solution to this problem can be deduced by examining the auto-correlation function (ACF) of the noise shaped part of the output signal. From equation 4.1.2.d, the output noise is known to be filtered quantisation noise, so the ACF of the output noise will be the convolution of the quantisation noise ACF and the NTF impulse response. If the quantisation noise can be assumed to be white, its ACF can be assumed to be a unit impulse at t=0, so the ACF of the output noise is simply a copy of the NTF. This is most significant since the non-zero value of the output ACF at odd increments of the sampling period implies that the noise in the output is correlated to at least part of the samples with the opposite vector. Put simply, noise modulating one edge is partly made up of noise quantised from opposite edges. This confirms the parasitic AM->PM which was discussed before. From this observation, the required NTF can be deduced.

Since the odd indexed samples of the ACF must be zero to ensure that no edge is made up of noise from an oppositely modulated edge, the odd indexed coefficients of the NTF must also be zero. Interleaving of the NTF with zero-valued coefficients restores the noise shaper / 2SCPWM unit to its expected characteristic (mildly non-linear). While interleaving the NTF with zeros might seem to be an elegant solution, it does have one drawback: the NTF develops symmetry about Fs/4 (ie. the signal band will be halved by interleaving). To get around this, the signal band should be designed to be twice

the bandwidth of a conventional NTF, consequently, the power gain will be higher. Implementing zero-interleaved noise shapers is not much more complicated than standard noise shaper since zero-valued coefficients need not be evaluated. Twice the register capacity is used in the zero interleaved structure to accommodate the lengthening of the filter, but otherwise the processing complexity remains the same. Figure 4.5.5.b below shows the typical structure.

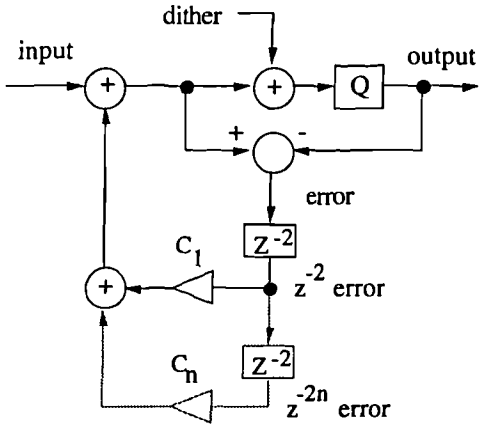


Figure 4.5.5.b : Modified Structure For a Noise Shaper Using a Zero-Interleaved NTF

In light of the fact that 2SCPWM uses a sample rate twice that of the pulse repetition rate (see section 2.2.3) a simple way of applying zero interleaved NTFs is to operate the noise shaper at twice the sample rate (as had to be done with the dual noise shaper architecture). While this does involve more interpolation, it is most probably after significant sample rate increase anyway, so the required filtering can be low order (see section 3.1.2). If this approach is adopted, the new Fs/4 coincides with the NTF's designed Fs/2; ie. the signal band after interleaving is the correct bandwidth in the doubled sample rate application and does not have to be designed to be excessively large. In many instances this proves satisfactory, however the NTF signal band suppression requirement should be revised because the requantisation noise introduced by the noise shaper is produced at twice the rate and hence spread with half the PSD value. Equation 4.3.3.i should be re-applied in this case, with the oversampling ratio, 'L', set at twice the design sample rate since the design after zero interleaving will be operating at this rate.

While two sided modulation types 2SCPWM and DSPWM both use vector data rather than scalar data and suffer parasitic AM->PM, SYMPWM is a special case which uses two edge modulation but does not suffer in this way. This is because the data has again become scalar, by incrementing both edges in equal and opposite amounts.

Introducing symmetry about Fs/4 makes general high-pass designs become band-pass designs. As shown in chapter 2, all PWM tends to modulate spectral replicates of signals near Fs/2 back to near DC (the audio band). Band pass filters are useful, even in single sided modulation types for reducing this problem (this will be looked at in the next section). Fortunately, although 2SCPWM is more sensitive to this kind of non-linearity than SYMPWM or AOAPWM (see figure 2.4.5.a), it is the very case which already requires bandpass filters implicitly as a result of solving the AM-PM problems.

4.5.6 : Band Pass NTF Design to Avoid Noise Sideband Distortion.

Although band-pass filters can be produced by zero-interleaving the NTF of a high-pass design for use with two sided PWM types, lower gain band-pass filters can be produced by optimising with two regions of interest, typically one near Fs/2, the other the standard signal band. These can be useful for reducing noise-carrier intermodulation which would otherwise re-appear in the signal band in the single edged PWM types, which are prone to this kind of intermodulation distortion (cf. figure 2.4.5.a).

PWM is known to harmonically modulate signals into sidebands of the pulse repetition rate (carrier), and the lowest harmonic indices have the largest amplitude (cf. equations 2.3.2a,3a,4a,5a & 6a). This would suggest that frequencies within half the signal bandwidth of Fs/2 will form sidebands with the potential (if they are not kept small) to destroy the linearity in the audio band. If shaped quantisation noise at these frequencies is allowed to be large, it may well reduce the signal band SNR substantially. This is not directly the consequence of inadequate noise shaping, but more the interaction of high power gain noise shaping and PWM's non-LTI behaviour.

The amplitude of the sidebands produced by the PWM unit varies with the modulation type, the harmonic index, the input signal amplitude, the oversampling ratio, and the input signal frequency. As a result of using oversampling, the problem only becomes significant in high resolution systems (>120 dB audio band resolution) or low wordlength PWMs, and usually with poor noise shaper NTFs (sinusoidal NTFs are particularly prone to this problem, see figure 4.2.1.a). Furthermore odd order, minimum-power-gain-optimised NTFs usually have a zero at Fs/2 assisting in reducing the amplitude in this sensitive region (see section 4.4.4 and figure 4.4.4.a).

As a rough gauge of the relative sensitivity of the high frequency bands, a programme was written (see appendix A.2.3) to plot the amplitude at a high frequency which introduced sidebands in the audio band above a given level; 'foldback.pas' operates by repeatedly calculating sideband amplitudes (by calling a high accuracy Bessel Function routine, 'Bessel.unt') for a given input frequency, and using binary search to locate the amplitude at which the sideband is at a given level in the audio band; an example of its output is shown below:

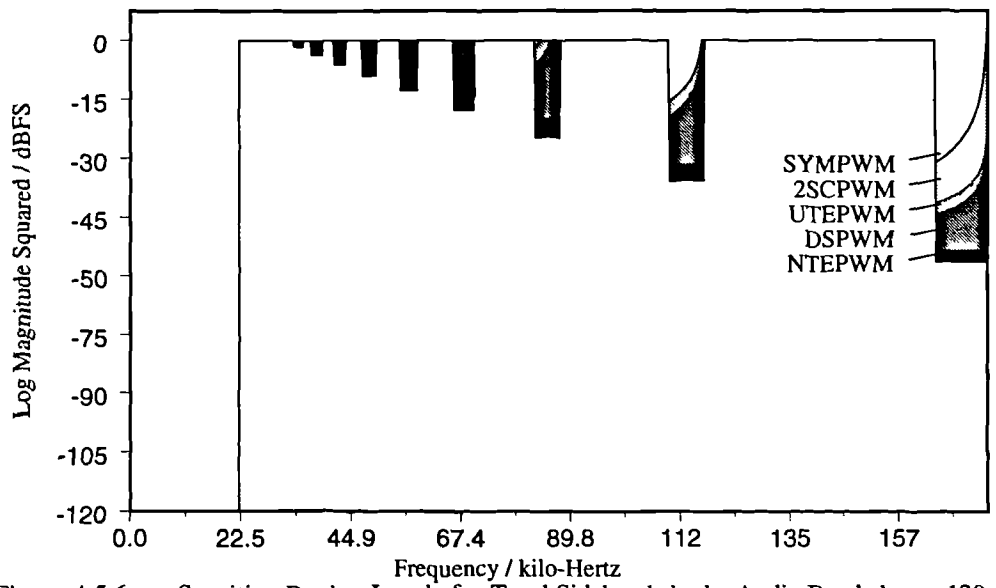


Figure 4.5.6.a : Sensitive Region Levels for Tonal Sidebands in the Audio Band above -120 dB

It should be noted that this is only an estimate of the sensitive region (in amplitude terms) because noise does not necessarily modulate in the same way as tones. Differences have been observed in simulation, but no conclusive results have yet been found (ie. useful enough to steer design techniques).

Since the problem is a non-LTI one, an analytic assessment of this problem is not appropriate (it is more of an optimisation problem), however the problem cannot be properly examined without incorporating a pulse width modulator into the NTF optimisation routines (which would make assessment of the NTF very slow due to the decimation required). To date, only checking the performance of NTFs by simulation has been found effective in designing for the above mentioned 'difficult' cases. An acceptable design can be approached iteratively knowing that relaxation of any one of the criteria will help avoid these problems.

4.5.7 : Multi-band NTF Design to Avoid 'Near-Frequency' Intermodulation.

Multi-band filters can be useful in reducing the effects of the PWM's non-LTI behaviour as an extension of the band-pass approach to avoiding 'sensitive' regions.

Not only does uniformly sampled PWM exhibit sideband behaviour, but also intermodulation distortion between each signal component. This is particularly relevant to DPWMs which cannot inherently use natural sampling (relying instead on some form of pre-compensation or error feedback) which modulate any uncompensated-for noise as if it were to be modulated as uniformly sampled PWM. As the uniformly sampled modulation types provide better sideband performance this is a good feature of pre-compensated systems (ie. they are less sensitive than real naturally sampled types). However, when large amplitude tones in the audio band are modulated (whether compensated for or not), they will intermodulate with the noise and this reduces the SNR in the signal band. Signals closer in frequency to the noise show this effect more than signals further away, and high amplitude signals are particularly bad (-10 to 0 dB). It is debatable whether signal masking would render this effect audible since the SNR reduction is about 10 dB with a 0 dB tone at 20 KHz (ie. worst case conditions). Shown below are the zoomed-in spectra from a noise shaped signal representing a 20 kHz tone at full amplitude, before and after pulse width modulation (figure 4.5.7.a). In the modulated case the rising floor around 20 kHz is an example of the worst case intermodulation noise which arises from using noise shaper NTFs which rise steeply just after the audio band (as can be seen in the input signal to the PWM).

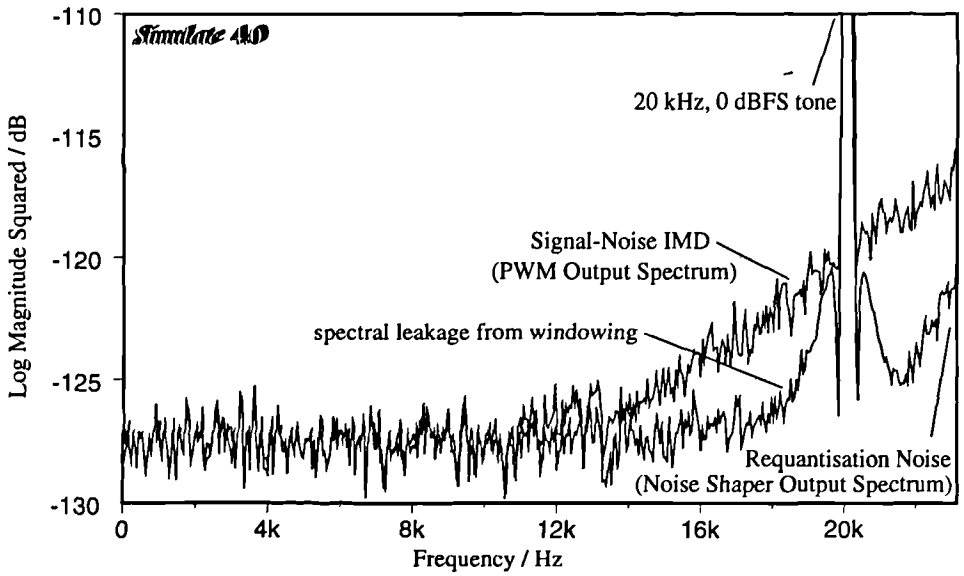


Figure 4.5.7.a : Signal-Noise Intermodulation, Before and After PWM

To reduce this effect, the NTF can be modified to have a buffer zone from 20 KHz up to around 30 KHz where the noise power is restricted. Programme 'mbnsopt.for' is ideal for this application since multiple bands can be specified with arbitrary gain and weighting.

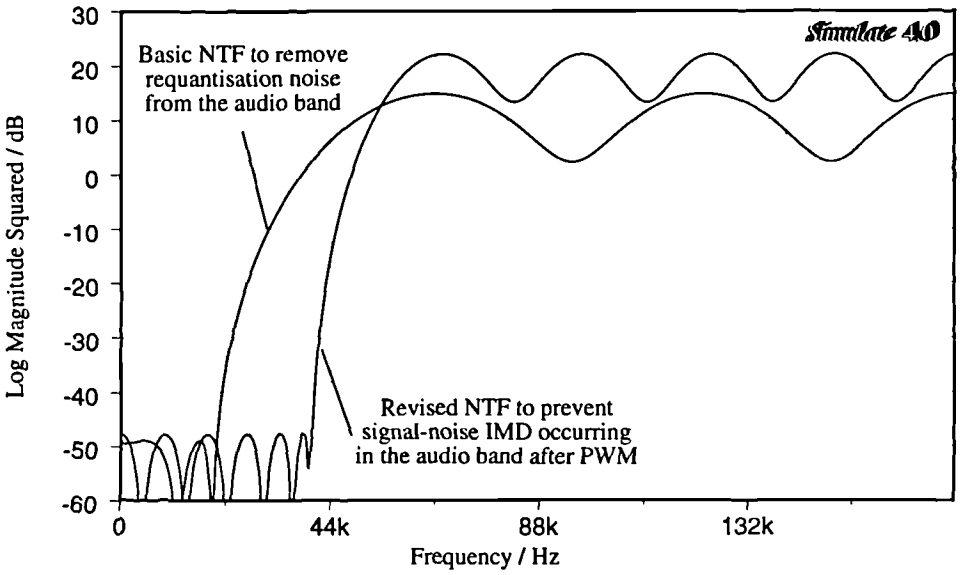


Figure 4.5.7.b : NTF for Signal-Noise Intermodulation Reduction

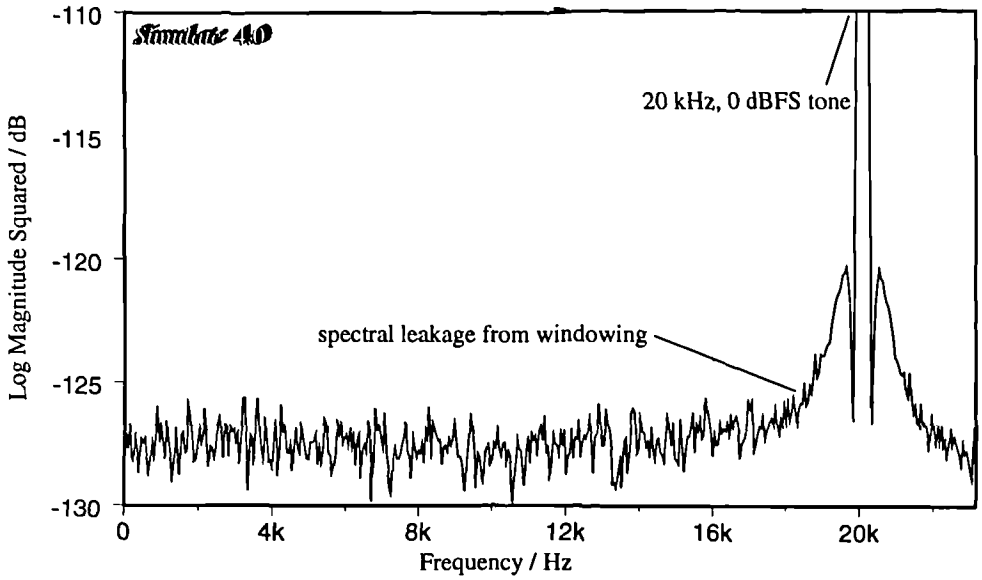


Figure 4.5.7.c : Reduced Signal-Noise Intermodulation

In light of the multiple frequency bands known to produce distortion for various reasons, multi-band optimisation is seen as the only way of improving current NTFs with a view to high resolution open loop PWM DACs (and maybe closed loop versions too). It would seem that the only rigorous way of ensuring minimum noise in the audio band with a particular set of parameters (input, output wordlength, oversampling ratio & SNR loss) would be to apply a global optimisation routine (eg. a simulated annealer) to work on a simulation of the system. Preliminary investigation of optimisation of a noise shaper's performance shows this to be a very slow process. When decimation filtering is added within each optimisation cycle, the rate of progress becomes prohibitively slow.

4.6 : Multi-Quantiser Noise Shapers.

In the implementation of higher order noise shapers (>2), modularity becomes an important feature which permits easy checking of circuitry. Where noise shapers are constructed of multiple low complexity blocks, design and testing is greatly simplified. This will be demonstrated in practice in section 6.2 where a fourth order noise shaper is constructed from two second order elements, but first, in this section, the theoretically required intermediate signals will be evaluated.

A natural consequence of using two noise shapers to build up a higher order system, is the introduction of two quantisers (and thus two sources of requantisation noise). Each quantiser must be adequately dithered to maintain linear performance, so to avoid adding dither noise on two occasions, structures have been developed in which the processed results of the two dithers cancel. The relationships required to ensure cancellation of the dither will be evaluated in this section, and the structures employed will be demonstrated by simulation. This is such a useful result that it could warrant a dual quantiser noise shaper in place of a basic noise shaper in spite of the additional complexity required.

4.6.1 : Analysis of Parallel Multi-Quantiser Noise Shapers.

When two or more noise shapers are put in series or parallel, there will be more than one quantiser, and hence more than on source of requantisation noise. Just as in digital filtering, a cascade of low order sections can be used to build up a high order system and in some cases this can ease the design by spreading the system complexity more evenly. Where noise shaping is cascaded, two particular structures are useful, the series form and the parallel form, named according to how the error signals are processed; each has its 'pros and cons'. The two systems will be examined separately since they offer different benefits and drawbacks, starting with the parallel form. This is based on the block diagram shown below:

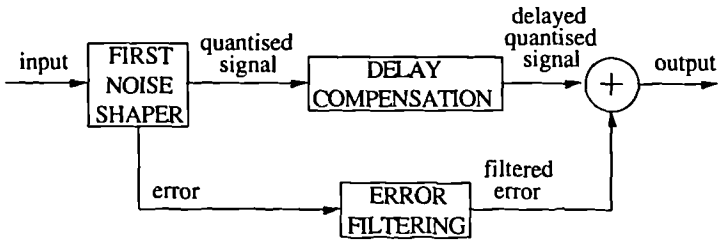


Figure 4.6.1.a : Block Diagram of a Parallel-Form, Multi-Quantiser, Noise-Shaper.

Taking a system made up of two first order sinusoidal noise shaper elements, quick analysis can be used to demonstrate the required error filtering and delay compensation networks to complete an equivalent second order system. Since a second order sinusoidal NTF is built up of two first order sinusoidal NTFs, multiplied together, this is also the expected overall NTF in this simple case.

From equation 4.1.2.b and 4.1.2.d, the output of the first noise shaper is known to be:

$$Q = I - (1 - H(z))\,E \qquad \text{Equation 4.6.1.a}$$

From this it follows that if the error signal, E, can be filtered (by (1-H(z)) and added back in

to the output, the filtered noise can be cancelled. While this is appropriate for cancelling the output noise, it will also destroy the output wordlength reduction. To avoid this the multiplied error signal must itself be noise shaped as below (see figure 4.6.1b).

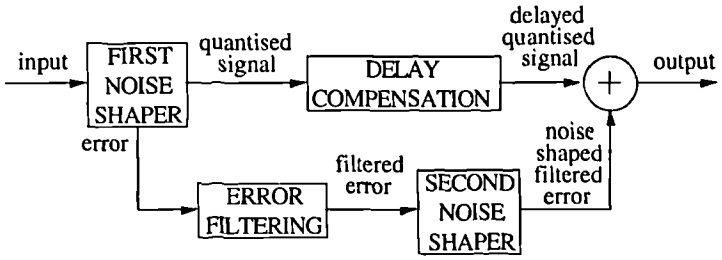


Figure 4.6.1.b : Error Filtering First in a Parallel-Form, Multi-Quantiser, Noise-Shaper.

Since the output from the second noise shaper is made up of its input plus some shaped requantisation noise, the error filtering and second noise shaper can be interchanged, allowing lower wordlength in the error filtering (see figure 4.6.1.c). This only applies to systems where the first noise shaper has an NTFs with integer coefficients as will be seen shortly.

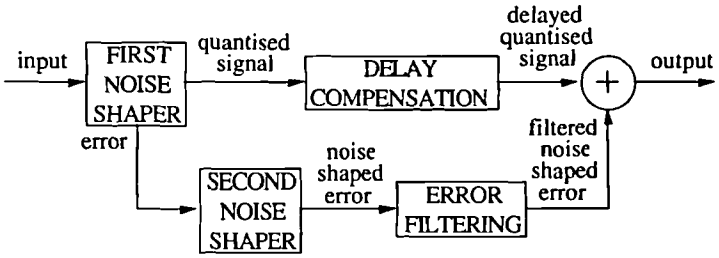


Figure 4.6.1.c : Error Filtering Second in a Parallel-Form, Multi-Quantiser, Noise-Shaper

If the second noise shaper comes after the error filtering (figure 4.6.1.b), the NTF of the overall system, becomes dominated by that of the second noise shaper alone. This order is not very useful since it can be achieved more efficiently in a single quantiser noise shaper

If the second noise shaper comes before the error filtering (figure 4.6.1.c), the filtering will increase the wordlength of the cancellation signal . In the special case that the coefficients of the first noise shaper's NTF are integeric (and thus the error filtering coefficients are also integeric) the wordlength of the cancellation signal does not extend before the binary point. In this case, the wordlength reduction of the output is not lost. Since sinusoidal NTFs have integeric coefficients they are particularly suitable for this parallel multi-quantiser structure.

Each noise shaper on its own does not theoretically introduce delay, but in practical systems where the input and output are latched while the current data is processed in between, a delay of at least one clock cycle is introduced in the signal path. This delay in the second noise shaper has to be compensated for by delay in the signal path for correct cancellation of the first shaper's error at the output adder. Thus the overall network for the simple example discussed is as shown below:

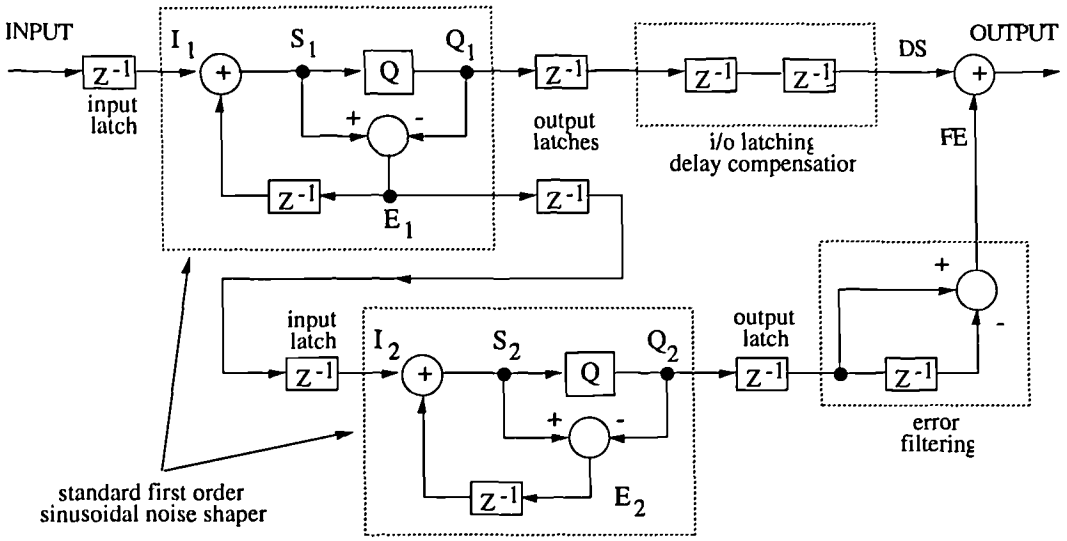


Figure 4.6.1.d ; Detailed Block Diagram of a Dual-Quantiser, First-Order, Parallel-Form, Noise-Shaper

Using the nomenclature as in above (figure 4.6.1.d), and the analysis technique of section 4.1.2,

$$I_1 = z^{-1}.\ INPUT$$

$$Q_1 = z^{-1}.\ INPUT - (1 - z^{-1}).\ E_1$$

$$I_2 = z^{-2}.\ E_1$$

$$Q_2 = z^{-2}.\ E_1 - (1 - z^{-1}).\ E_2$$

hence the filtered error, $FE = (1 - z^{-1}).\ z^{-1}.\ \{z^{-2}.\ E_1 - (1 - z^{-1}).\ E_2\}$

$$= z^{-3}.\ (1 - z^{-1}).\ E_1 - z^{-1}.\ (1 - z^{-1})^2.\ E_2 \qquad \text{Equation 4.6.1.b}$$

after delaying Q1, $DS = z^{-4}.\ INPUT - z^{-3}.\ (1 - z^{-1}).\ E_1 \qquad$ Equation 4.6.1.c

hence the output, $FE+DS = z^{-4}.\ INPUT - z^{-1}.\ (1 - z^{-1})^2.\ E_2 \qquad$ Equation 4.6.1.d

This output can be seen to comprise two parts, the delayed input, and the delayed filtered quantisation noise. It is also worth noting that the NTF is that of a second order sinusoidal noise shaper and the noise in the output is not a function of the quantisation level chosen for the first noise shaper at all but rather the second noise shaper alone.

The operation of such a system, but based on second order sinusoidal noise shaping elements, has been simulated as programme 'dcns4.pas' and constructed in a combination of ASIC and TTL series CMOS (see section 6.2). The performance of the two is very similar; the software output is shown below:

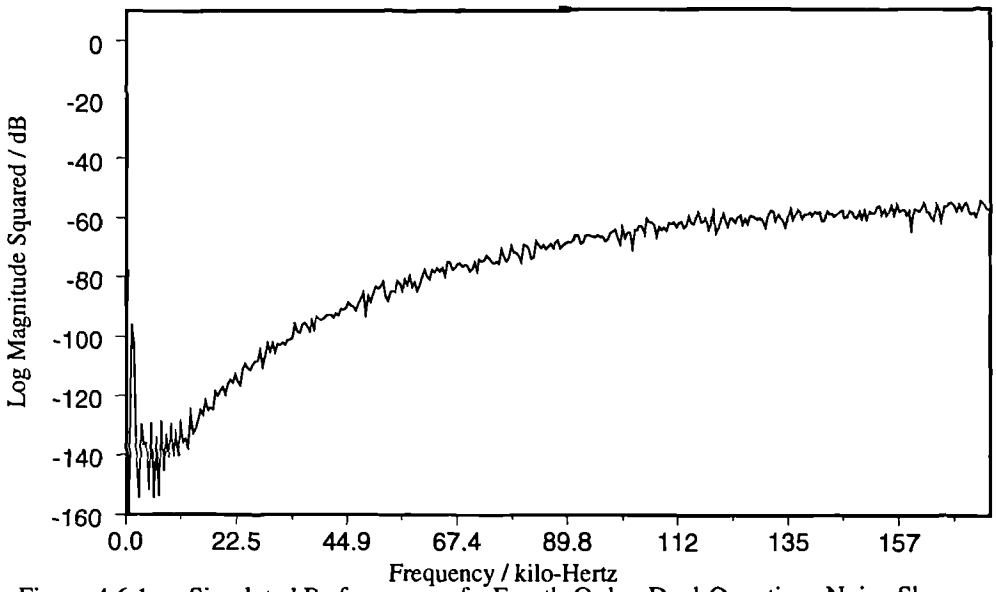


Figure 4.6.1.e : Simulated Performance of a Fourth-Order, Dual-Quantiser, Noise-Shaper.

4.6.2 : Dithering a Parallel-Form Multi-Quantiser Noise Shaper.

Having two quantisers introduces the need to dither each appropriately. Having two opportunities to dither the overall shaper brings the opportunity to use self cancelling dither, but the system becomes more complicated than at first glance.

From equation 4.6.1.b, where the input was known to be the delayed error from the first shaper, the error processing can be seen to be a multiplication by (1-H(z)) and a delay of 2 sample periods. From equation 4.6.1.c, the signal processing can be seen to have merely delayed the signal by 4 sample periods. If 'node B' dither (cf. section 4.1.5) is used in both cases (and this sort of dither is known to have the same transfer function as the signal), the two dithers can be constructed to cancel if:

$$z^{-4}.\ D_1 = z^{-2}.\ (1 - z^{-1})\ .\ D_2$$

or put more generally, $z^{-2}.\ D_1 = (\ 1 - H(z)\ )\ .\ D_2$ Equation 4.6.2.a

If each were adequately sized this would appear fine, except that for small signals (precisely when dither becomes a necessity) almost all the input (and dither) of the first noise shaper is passed to the second. Subsequently, the second dither is added to this input stream to the second noise shaper, cancelling before it reaches the second quantiser, leaving the second noise shaper inadequately dithered. For these reasons, additive dither would seem the most useful way to dither the parallel form of multi-quantiser noise shaper, however more research should be done to establish whether larger amplitude subtractive dither following, 4.6.2.a, could be used to overcome these problems.

4.6.3 : Series Form Multi-Quantiser Noise Shapers.

Putting two noise shapers in series implies that both the signal and the shaped quantisation noise from the first shaper are treated as the input to the second shaper. This means that the output of the system becomes the signal plus the shaped requantisation noise of both shapers; generally this will be larger than using one noise shaping loop with an NTF which is the product of the two separate NTFs. The general structure of such a system is shown below:

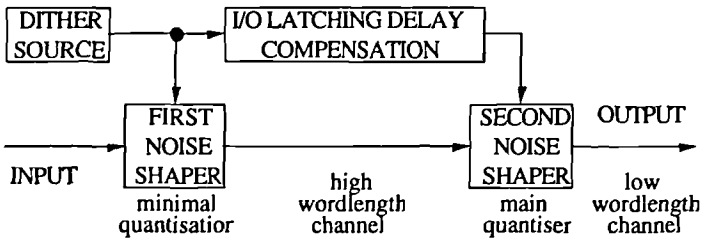


Figure 4.6.3.a : Series-Form, Dual-Quantiser, Noise-Shaping.

Since the quantisation levels of the two shapers must be different (otherwise the second shaper would not be brought into action at all), the requantisation noise added by each will be at different levels. By using the first noise shaper operating at a low level, and a second shaper completing the main quantisation, two opportunities to dither the channel exist, and self cancelling or 'subtractive' dither can be applied [ROB62]. The detailed structure of such a system is shown below :

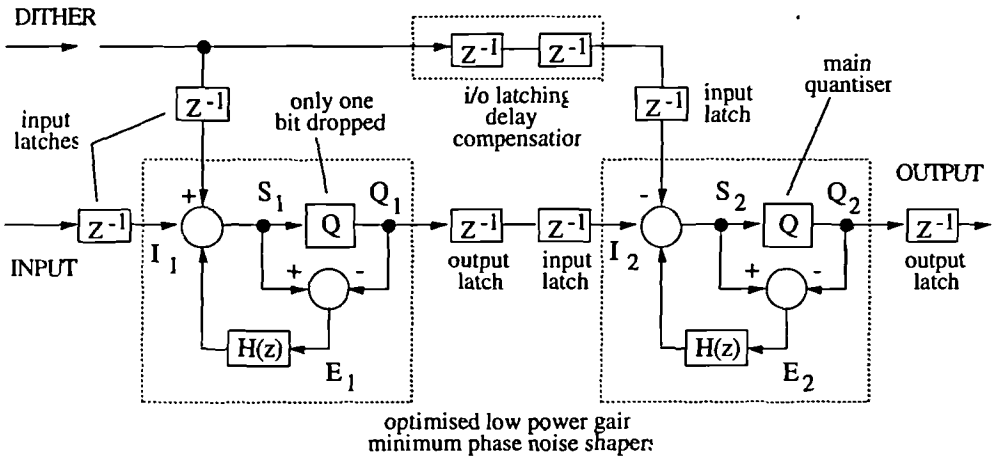


Figure 4.6.3.b : Subtractively-Dithered, Series-Form, Multi-Quantiser, Noise-Shaping.

As in section 4.6.1, noise shapers for all practical purposes incorporate delay, and this should be taken into account in the preparation of the dither to ensure it cancels correctly. One interesting point to note about this structure is that the second shaper would appear not to be adequately dithered. This is in fact not the case. The noise generated by the first noise shaper is left to dither the input to the second shaper. The first noise shaper can have a very low gain NTF to force the shaped requantisation well out of band since the signal band suppression need only be mild - it drops very little resolution. Presuming appropriate additive dithering conditions have been applied to the first quantiser, this shaped noise is not correlated with the remainder of the requantisation noise added by the second. In

this way the originally added dither can be removed from the channel, leaving uncorrelated noise to dither the second (main) quantiser with noise power gain due to dithering approaching unity (this will be dependent on the NTF used in the first shaper).

In light of the minimum phase NTFs that can be produced by optimisation, tending to unity power gain for increasing order, this kind of dithering can be applied over and over again with very little penalty in cumulative noise. This has particular applications in long signal processing chains, especially multi-processor signal processing, where individual stages may be operating with floating point precision, but the signal has to go through repeated requantisation between processors. A simulation of the performance of a two stage multi-quantiser noise shaper is shown below, and although this uses sinusoidal NTF, the advantages are obvious in the audio band. 16->15 bit reduction by first order sinusoidal NTF is used in the first shaper, followed by 15->8 bit reduction by fourth order sinusoidal NTF in the second.

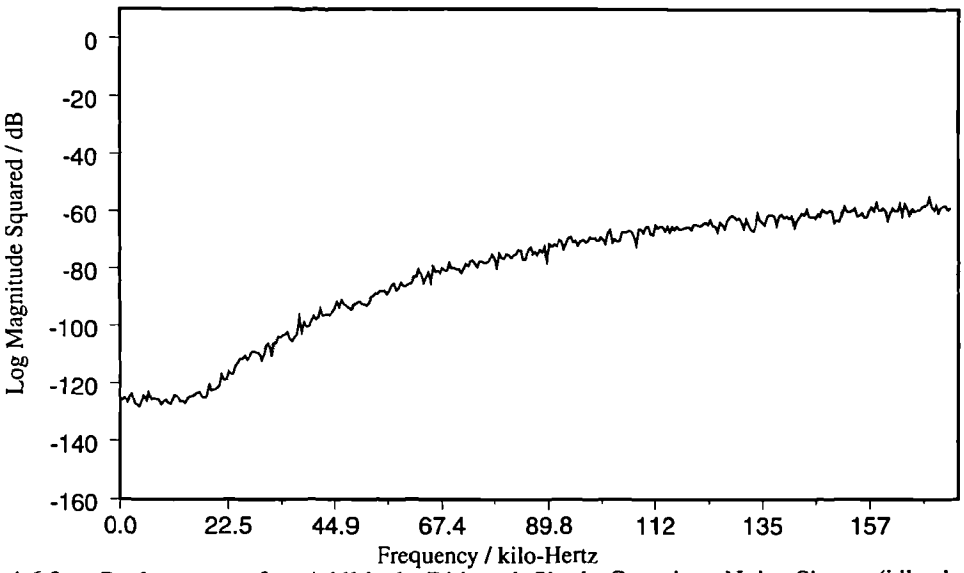


Figure 4.6.3.c : Performance of an Additively-Dithered, Single-Quantiser, Noise-Shaper (idle channel)

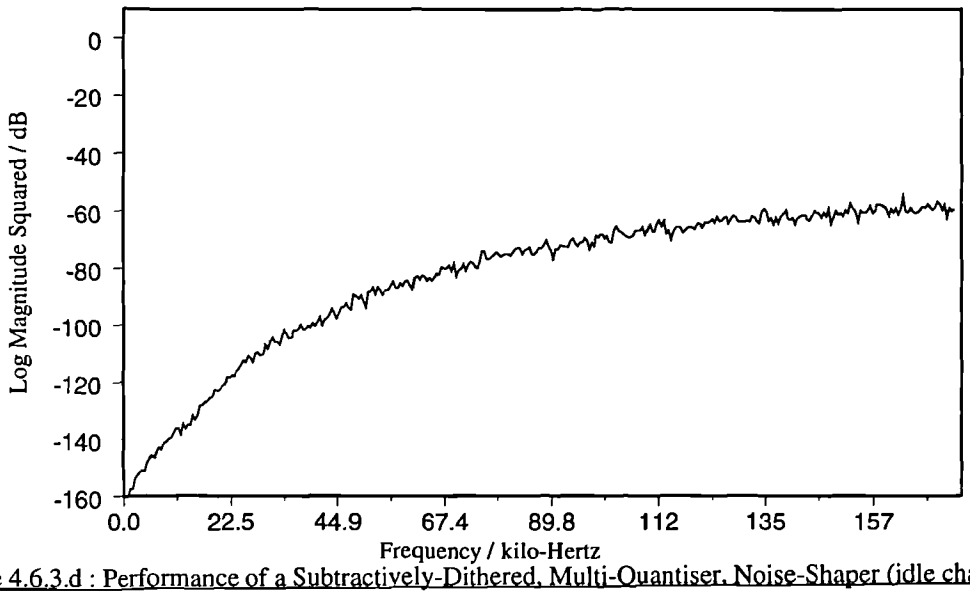


Figure 4.6.3.d : Performance of a Subtractively-Dithered, Multi-Quantiser, Noise-Shaper (idle channel)

4.7 : Summary.

Noise shaping can be used for wordlength reduction of a digital signal, allowing a high resolution signal to be represented in a shorter wordlength by forcing its lower bits to zero. The penalty to be paid for doing this is that the total noise power in the signal is increased. The noise signal which is added to force the lower bits to zero can be filtered to give it a small amplitude at frequencies of interest, thus leaving at least some of the available frequency band with a large signal to noise ratio.

In order to place the added noise at (high) frequencies which do not matter, spare bandwidth is required; this can be provided by oversampling or interpolation. The spectral characteristics of the added noise are related to the filtering used in the shaper. With knowledge of the number of input and output bits, the oversampling ratio and the required output signal to noise ratio, the filter's signal band requirements can be calculated (equation 4.3.3.i). Filters based on high order differencers are common ('sinusoidal'), but unnecessarily amplify the noise which is added to the output.

Minimum noise power gain can be guaranteed for a given output power spectrum shape by using a filter with minimum phase-lag properties. The minimum gain value has been seen to satisfy the requirement that the integral of the logarithm of the magnitude-frequency response is zero and from this filter shapes can be estimated. Minimum phase-lag filters can be generated by linear phase design followed by conversion to minimum phase, or by optimisation for minimum gain. Optimisation has the advantage that is constructs unspecified parts of the frequency response.

The minimum phase-lag filter has been seen to be a special case of a general philosophy : the effect of an error is minimised if it is corrected for as soon as possible. From this, gain minimisation in the filtering is seen to be directly related to delay minimisation. Thus, minimising the delay in filters with partially specified frequency response defines the shape of the unspecified frequency response. This filter will also have the minimum gain of all possible filters which satisfy the partial frequency response specifications.

Feedforward filters have been shown to be a special case, further reducing delay in the error filtering network by spreading requantisation error both *before and after* the sample from which it originates. Special filters with integer feedforward coefficients have been shown to reduce the output noise further than purely recursive shapers can achieve, almost to the information theory limit.

For small amplitude input signals the noise shaper output noise becomes significantly correlated to the input, introducing signal distortion for small signals and periodic output for a fixed (DC) input. Addition of a small amount of uncorrelated noise, 'dither', can be used to destroy these effects. Quantisation can be performed by truncation or rounding, where rounding yields a 3 dB lower total noise power. Several structures have been examined which use multiple quantisers to carry out the truncation or rounding; these yield the opportunity to dither the overall shaper with very little noise penalty and minor additional system complexity.

Chapter 5 : Digital PWM DAC Structures & Performance

Chapter Overview.

In this work so far, a DAC based on PWM has been assumed, the performance of PWM types has been examined, and a tentative structure as shown in the introduction (figure 1.1.1.a) has been worked towards. Knowing that digital PWM can only operate with limited resolution despite its high accuracy, the preceding DSP blocks have been designed to enable CD quality signals to be passed to the PWM in a form it can handle. Specifically, noise shaping has been used to reduce data wordlength, and interpolation has been used to provide the excess bandwidth required for noise shaping (this also reduces the distortion produced by the PWM slightly).

The problem remains : how can the PWM based DAC be made to distort the signal less, when PWM itself is not a linear, time-invariant process ?

In this chapter, alternative architectures will be examined [†], one from the author, the others from the literature. Five basic strategies emerge, three based on observation of PWM's behaviour and using an open-loop correction strategy, and two based on the well known concept of negative feedback as commonly used in analogue power amplifiers : ie. a closed-loop correction strategy. These will be examined in the following order :

1) Simple, open-loop architectures - systems where the choice of modulation type and the modulation parameters are chosen for best performance.
2) Static, pre-compensation based architectures - again, systems operated in an open-loop structure, but now with the addition of a time-invariant pre-compensation scheme. Most of these systems derive their improved performance by approximating a better sampling type.
3) Time-variant, pre-compensation based architectures - still, systems operated in an open-loop structure, here with an adaptive (signal dependent), time-variant pre-compensation.
4) Closed loop architectures, using output decimation - systems using error feedback or signal feedback where the output of the PWM is decimated (real time) and errors are reduced by using negative feedback with a high loop gain.
5) Closed loop architectures, using output emulation - systems where the output is either predicted or the output error is emulated (or both) so that real time decimation can be avoided.

[†] Several of the techniques described in this chapter appeared in the literature during the period of this work. Parts of these may not be discussed until after some of the contributions from the author. Although this is recognised as an unorthodox approach, it is hoped this will help the reader to establish a good understanding of the basic systems before tackling the more advanced techniques.

5.1 : Open Loop PWM DAC Architectures.

5.1.1 : The Basic Structure : Variables and Constraints for PWM DACs.

In its most basic form, a PWM DAC can be constructed with three processing elements : pulse width modulation to perform digital to analogue conversion, noise shaping to reduce data wordlength at the input to the PWM, and interpolation to provide excess bandwidth for the noise shaper. To these basic processing elements, an output amplifier can be added (to increase the amplitude of the output signal) and a recovery filter is usually used. The output amplifier can be of class D type (switching between totally off, and 'saturated' on') and thus highly efficient but is not intended to modify the signal in any other respect. The recovery filter is usually of low pass type, passive, and should not interfere with the reproduction of the audio band signals. To avoid small signal non-idealities in the noise shaper, a dither source is also required. To summarise, the basic architecture is as shown below :

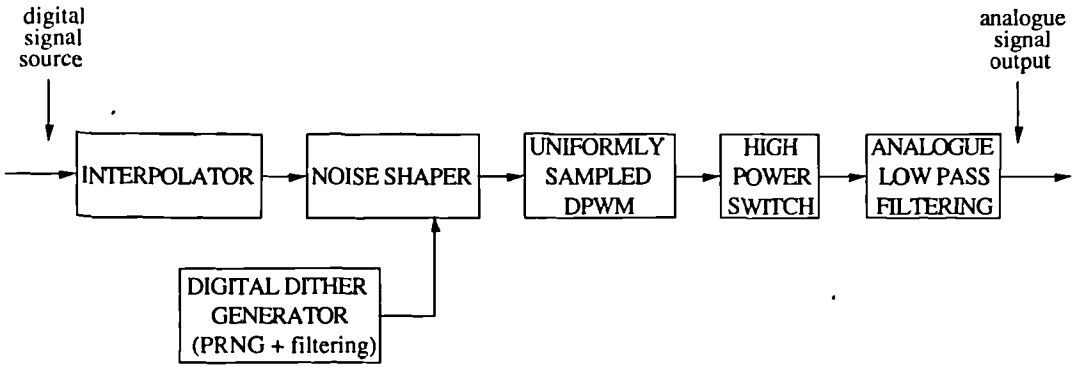


Figure 5.1.1.a : Basic PWM DAC Structure

From chapter two, it is known that reducing the modulation depth within the PWM reduces the distortion it produces. Similarly, decreasing the ratio between the signal and carrier, ω_v/ω_c, reduces the distortion produced by the PWM. Furthermore, various PWM types are available, some with better harmonic performance than others, and all with minor sideband and intermodulation problems. For a DAC in its simplest form as shown above, these are the only parameters which can be varied to improve performance.

Overall the DAC is constrained by 5 design objectives, which in order of significance are:

1) The system must be one which can be implemented using commercially available devices,

2) The output SNR should be as high as possible,

3) The output THD should be as low as possible,

† 4) The output efficiency should be as high as possible,

† 5) The output power should be at least hundreds of milliwatts, preferably hundreds of watts.

Satisfying (1) mainly concerns implementing the PWM unit itself. 'Off the shelf' discrete logic families can currently operate at up to 60 MHz system speed (74AC family) which with careful circuit design can be extended to 100 MHz. This will be regarded as the maximum clock rate at the moment.

† Constraints (4) & (5) are more relevant to power DAC design which is tackled in chapter seven.

(1) and (2) can largely be satisfied by appropriate choice of wordlength and oversampling ratio so that a noise shaper with low enough internal clock rate and an adequate signal band resolution is used. 'Fine tuning' of the noise shaper can then be applied to reduce the slight noise intermodulation and noise sideband problems as discussed in section 4.5 .

Unfortunately, objectives (3) & (4) are in conflict since reducing the oversampling ratio to increase the output efficiency also increases the distortion produced by the PWM. Given that some compromise will be made between (3) and (4), the choice of oversampling ratio is limited by objective (5), because as the output power is increased the linearity requirements may take on a lower priority than the output efficiency (see section 1.5.1). To summarise, the first system variable to be chosen will be the oversampling ratio.

The oversampling ratio choice can be clarified by taking a close look at the form of the distortion introduced by the basic PWM types (two level output types); from chapter two the distortion in uniformly sampled modulation types is known to be made up of two elements : harmonic distortion and signal-carrier sidebands. Similarly, naturally sampled modulation types are known to produce only sideband distortion (although at a higher level than the uniform's sidebands). In both cases the sideband performance cannot be avoided other than by oversampling, so this should be used to define the lower limit of the carrier frequency. If increasing the oversampling ratio is to be used to eliminate harmonic distortion as well, this lower limit has to be raised a lot further when the uniformly sampled modulation types are used.

Five uniformly sampled modulation types can be constructed by appropriate modulator design; these are : TEPWM, LEPWM, SYMPWM, AOAPWM and 2SCPWM. Alternative sampling types (which are more like natural) require additional preprocessing and are discussed in section 5.2. From the theoretical spectral performance given in each section of chapter two, and assuming a worst case signal, the carrier frequency limits can be tabulated; however, as will be shown shortly, oversampling alone cannot be used to solve PWM's distortion problems.

For CD applications, a 20 kHz signal band is assumed, represented by a 16 bit signal sampled at 44.1 kHz. Other applications may be more or less stringent but the same approach can be used.

From the bandwidth, the worst case signal frequency can be calculated depending on the distortion structure. In all the modulation types the distortion increases with frequency and signal amplitude, and the largest distortion is the lowest harmonic (ie. this is the dominant component of the THD measurement). In most of the modulation types, the worst harmonic is the second, thus for this to occur at the limit of the signal band the worst signal frequency will be 20 kHz/2 (10 kHz). 2SCPWM only produces odd order distortion so the most significant harmonic is at 3 times the signal frequency and the worst case signal is at approximately 20/3 kHz (6.6 kHz). With these test tone frequencies and using a 'realistic' amplitude signal (0.1 full scale) the minimum oversampling ratio can be found for an acceptable level of THD. The maximum permitted level of in-band THD was chosen at 0.01%, since this is comparable to a peak SNR of 100 dBFS : the best that can be achieved in a 16 bit channel. The choice of signal amplitude and distortion level is arbitrary, chosen in this case to demonstrate performance for a realistic 16 bit signal with similar levels of distortion and noise.

Mod'n Type	Test Freq.	Distortion @	Intervals/pulse	Samples/pulse	Minimum Fc	Minimum Fs
TEPWM	10 KHz	20 KHz	2^b'	1	15.1 MHz	15.1 MHz
LEPWM	10KHz	20 KHz	2^b'	1	15.1 MHz	15.1 MHz
SYMPWM	10 KHz	20 KHz	2^(b'+1)	1	1.54 MHz	1.54 MHz
AOAPWM	10 KHz	20 KHz	2^b'	1	1.54 MHz	1.54 MHz
2SCPWM	6.6 KHz	19.8 KHz	2^b'	2	124 KHz	248 KHz

Figure 5.1.1.b : A Table of Minimum Carrier Frequency for Acceptable Distortion Across PWM Types
(b' is the input wordlength to the PWM, the same as the output wordlength from the noise shaper)

From this table, the minimum carrier frequencies can be seen to be very high for the single sided PWM types (TEPWM, LEPWM). If the modulator can only be clocked at 100 MHz (max.) these modulation types can only support 6 or 7 level PWM. Noise shaping down to so few bits is very difficult since the shaped requantisation noise swamps the signal, so these modulation types will not be considered further for a basic PWM DAC.

For the other PWM types, the wordlength for the various carrier rates can be calculated. This is straightforward in the AOAPWM cases, since the internal speed of the PWM is simply the product of the carrier frequency and the number of output levels, ie. Internal clock frequency = $F_c \cdot 2^{b'}$

In the case of SYMPWM the output levels must be exactly symmetric about the regular timing marker. This requires twice the internal clock frequency of an AOAPWM because of the need to represent odd valued samples which split to have pulse edges that should align to half 'time-quanta'.

In the case of 2SCPWM, two samples have to be represented in one carrier period, so the wordlength in each sample has to be one bit less of that used in an AOAPWM.

In a totally flexible system, the noise shaper and PWM sample rate can be chosen to be any ratio of two integers multiplied by the input sample rate by using excessive interpolation followed by decimation. For practical purposes a standard ratio (eg. 2^4 or 2^8) can be used with asynchronous sample rate conversion being used to 'make up the difference' [ADI93].

In this application the PWM is known to operate on binary data, so the number of output levels is always a power of two. Dividing the maximum counter speed by the ideal sample rate leaves a number which can be rounded down to the nearest number of output levels (which is assumed to be a power of two). By dividing the 100 MHz limit by the number of output levels, a maximum sample rate can be established. From this the oversampling ratio is chosen and the actual clocks, calculated.

Mod'n Type	Req'd Fs	Max No. Bits	Max. Fs	Min OSRatio	Implied Fs	Implied Clock
SYMPWM	1.54 MHz	5	1.6 MHz	36	1.588 MHz	† 101.6 MHz
AOAPWM	1.54 MHz	6	1.6 MHz	35	1.545 MHz	98.78 MHz
2SCPWM	248 KHz	8	390 kHz	6	264.6 kHz	67.74 MHz

Figure 5.1.1.c : A Table of Possible Sample Rates For a CD Quality PWM DAC

Noise shapers for the above sample rates and wordlengths were designed by optimisation and estimated for the sinusoidal case as in figure 4.2.2.b. SNR of the final output was measured by simulation for systems based on these filters, used at the shown sample rates and followed by the listed modulation types. Results for these are tabulated below.

† Clock speeds in excess of 100 MHz have been considered through rounding error in the calculations.

Mod'n Type	Fs	No. Bits	Clock Speed	NS Gain	THD / %	SNR / dB
SYMPWM	1.588 MHz	5	101.6 MHz	5.9719	0.00090	78.57
AOAPWM	1.545 MHz	6	98.78 MHz	3.2940	0.00106	78.30
2SCPWM	264.6 kHz	8	67.74 MHz	498.12	0.01148	57.35

Figure 5.1.1.d : A Table of Output SNR From Various PWM DACs Counting at up to 100 MHz .

These examples (tables 5.1.1.b, c & d) demonstrate the high level of distortion arising from † using oversampling alone to solve the PWM's linearity problems. The two single sided modulation types prove totally unsuitable for 'CD quality' conversion, even when operated with moderate testing conditions. With higher level signals, and lower minimum acceptable distortion requirements, even the symmetric types cannot be implemented in the basic open loop configuration. Only 2SCPWM shows potential, although for tougher tests, this too requires larger oversampling ratios than simulated above which are far from desirable for high power applications.

In light of the general inability of the basic open loop elements to provide the required quality, pre-compensation has to be sought to reduce the PWM distortion and hence avoid such large oversampling ratios. The structure for this will be looked at in section 5.2.

5.1.2 : Guard Bands, Pulse Borrowing & Gearing.

As well as designing to avoid large oversampling ratios, two other factors should be considered for simple PWM DACs. Pulse-pulse interaction can introduce distortion under large signal conditions and PWM overload arising from shaped noise can produce large distortion. Both of these can be solved by the introduction of a 'dead-time' between pulses which will be called a 'guard band'. The duration of the guard bands is largely dependent on the slew rate of the output devices. Typical values are an order of magnitude higher than the turn-on and turn-off times of the output devices and so in the range 50 - 150 ns. This is best found empirically and is discussed for a practical case in section 6.5 .

Guard bands avoid the pulses ever overlapping or forming a 'non-return-to-zero' signal. This time can also be borrowed from by pulses representing samples with large shaped noise content (ie. exceeding the normal range of the signal). Such practice avoids overloading the PWM, and will be called 'pulse borrowing'. The pulse separation can save gross distortion at the onset of PWM overload - a behaviour which is particularly important in power amplifiers. This technique is especially useful in high order sinusoidal noise shapers which have a large gain at Fs/2 (ie. alternating polarity requantisation noise samples).

To maintain maximum efficiency in a class D output stage (following the PWM) the carrier frequency should be as low as possible. For high amplitude, high frequency signals the oversampling would at a minimum anyway as a result of the design stages as described in the last section (5.1.1). For smaller signals or lower frequency signals the carrier frequency can be reduced without producing measurable distortion. Reduction by factors of two is common in the control of heavy-duty motors where efficiency is of more concern than harmonic content of the output; in that application this technique is called 'gearing' so the same name will be used here. In a more general sense, variable frequency PWM could be employed leading to a PDM/PWM hybrid which could operate with high accuracy, linearity and efficiency as a result of this extra degree of freedom.

† ie. the basic distortion is so big that very high pulse repetition rates have to be used (such as those shown) before reasonable THD+N and counting rates co-exist. This high PRR then limits their usefulness.

5.2 : Open Loop Architectures Using Time Invariant Pre-compensation

Section Overview

In this and the next section (5.3), systems for pre-compensating (or pre-distorting) the signal before the PWM will be discussed. This section will deal in detail with systems using time invariant pre-compensation, all of which attempt to approximate a better form of sampling than uniform sampling which was used in the basic PWM DAC described in the last section.

As seen in section two, natural sampling offers distortion performance with zero harmonic content and it is the proposal of this author that systems imitating natural sampling should be used to produce low distortion PWM based DACs [SAN91b,SAN92].

Three basic forms of pre-compensation have been found to be useful, with varying degrees of distortion reduction, and computational complexity. These will be called

1) 'Weighted Averaging' (for which a practical example is built in chapter 6)

2) 'Enhanced Sampling' - a pre-compensation technique from the literature [MAL67], and

3) 'Pseudo Natural Sampling' - as proposed by this author [HIO89] and expanded in [GOL92].

These will be investigated shortly, but first the architecture in which they operate must be chosen.

5.2.1 : The Architecture.

Pre-distortion (or 'Pre-compensation', as it will be referred to) can be placed in one of three places in the open loop PWM DAC structure of figure 5.1.1.a :

i) just before the PWM - this is the obvious place to counteract the action of the PWM,

ii) just before the noise shaper - this cannot compensate for noise added by the shaper,

& iii) at the input to the system - this has the advantage that is does not have to be operated as fast as the other two positions require (the sample rate is lower).

Putting compensation in just before the PWM is the obvious place to create a combined unit which can be assumed linear, but this has the drawback that it may well increase the wordlength of the signal, destroying the benefits of the noise shaping. To avoid this, the compensation should be placed before the noise shaper, but this implies that the shaped requantisation noise will not be compensated for and will be modulated as before (ie. as uniformly sampled). Since the requantisation noise is usually shaped to high frequency, where sideband performance is more of a concern than harmonic performance, this is not entirely a bad situation to have. The sideband performance of a uniform modulator is better than that of a naturally sampled modulator as was shown in chapter two, so in fact this order of processing elements allows the best of both sampling types. The revised block diagram would be as shown below (cf. figure 1.1.1.a) :

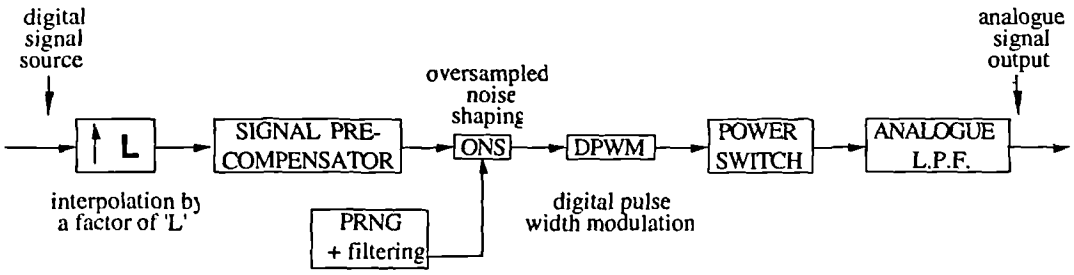


Figure 5.2.1.a : Pre-compensated PWM DAC Structure.

Placing the pre-compensation after the interpolator ensures that the signal is only slightly different, sample to sample. In general this makes the task of pre-compensation simpler as will be shown shortly, but this placing forces the pre-compensation to function 'L' times faster (where 'L' is the interpolation ratio) and interpolator errors can still upset the pre-compensation system.

The method of generating naturally sampled PWM was discussed in chapter two, and involved finding the end time of the pulse where the signal matched a sawtooth. Since the sawtooth amplitude is simply a linear function of time within one pulse period, this can be reduced to finding the underlying value of the signal at the end of the pulse. Thus, to imitate natural sampling, a pre-compensator which can find the value of the signal between samples is required. This is exactly what an interpolator does at regular intervals, although the pre-compensator is looking for points on a far finer time scale. Since these two functions can be considered the same, they could be integrated into one block, eliminating errors from the interpolator, as shown below :

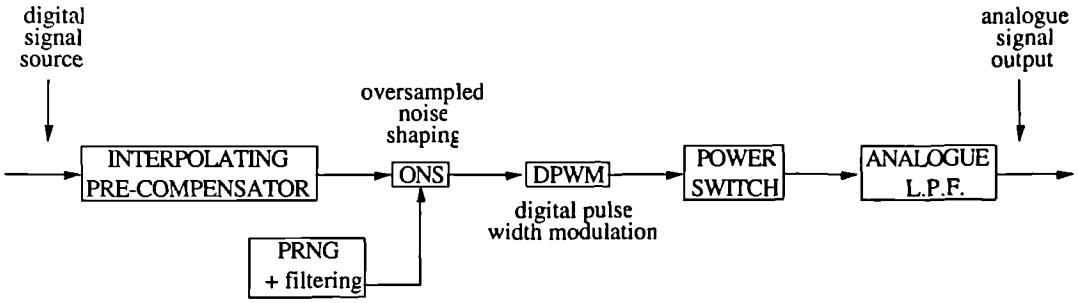


Figure 5.2.1.b : Non-linear Interpolator - PWM DAC Structure

Replacing the 'L'-times interpolator with a block which generates 'L' pre-compensated amplitudes means that the function has to be known considerably better (ie. more support abscissae are required). Although this technique cannot be upset by interpolator errors, it introduces errors of its own. The computational complexity of the algorithm to find the pre-compensated points turns out to be more than 'L' times that used at the high rate (ie. on the interpolated signal) [PAU92b]. Since interpolation is commercially available in simple blocks, DSP programming of this technique was not considered worthwhile. Alternative algorithms may be found to operate more efficiently, typically based on non-linear interpolation [GIN91] and could form the subject of future research.

To summarise, the best position of the three considered viable for pre-compensation of the signal in the open loop structure was chosen to be between the interpolator and the noise shaper. This architecture will be assumed for the rest of this section (5.2).

5.2.2 : Pre-compensation Algorithms : Sample Averaging

Natural sampling was observed in chapter two to have good distortion behaviour, and was produced in analogue circuitry by comparison with a sawtooth signal. If regularly sampled PCM data can be replaced with ideal pulse durations the distortion performance from the PWM DAC is expected to be better. Since any low frequency signal must intersect with the sawtooth between samples, it would be fair to assume that the average of samples either side of the pulse in question would be a closer estimate of the actual naturally sampled value.

This has been simulated and in fact provides a small distortion reduction (nearly 2 dB for the third harmonic under worst case distortion conditions). While in itself not very significant, this technique points the way to more sophisticated systems for distortion reduction. Also, this average value provides a good starting point for iterative estimates used in some of the techniques discussed later in this section (5.2.5, 5.2.6 & 5.2.7).

5.2.3 : Pre-compensation Algorithms : Weighted Averaging.

A closer approximation than taking the simple average is to use a straight line between samples and estimate the intersection time of this with a sawtooth. The new data value (which will be counted as a pulse duration by the DPWM) is known as the weighted average of the two adjacent samples and this kind of pre-compensation will be referred to as weighted average pulse width modulation (WAPWM). A diagram of this procedure is shown below along with the alternative end of pulse times.

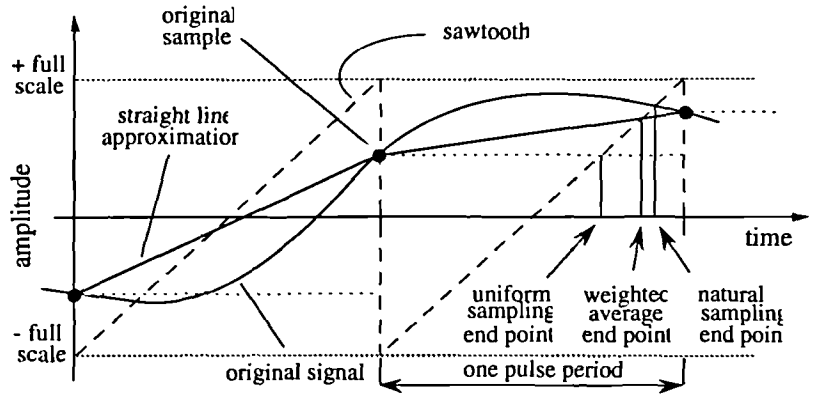


Figure 5.2.3.a : The Uniformly Sampled, Weighted Average and Naturally Sampled Intersection Points

Considering a single sample period in detail, the amplitude at the intersection point can be evaluated, by similar triangles. One example, taking sixteen bit input data and using 88% modulation depth is shown below to assist with the analysis which follows. These parameters have been chosen specially to be consistent with those used in hardware (see section 6.5.4).

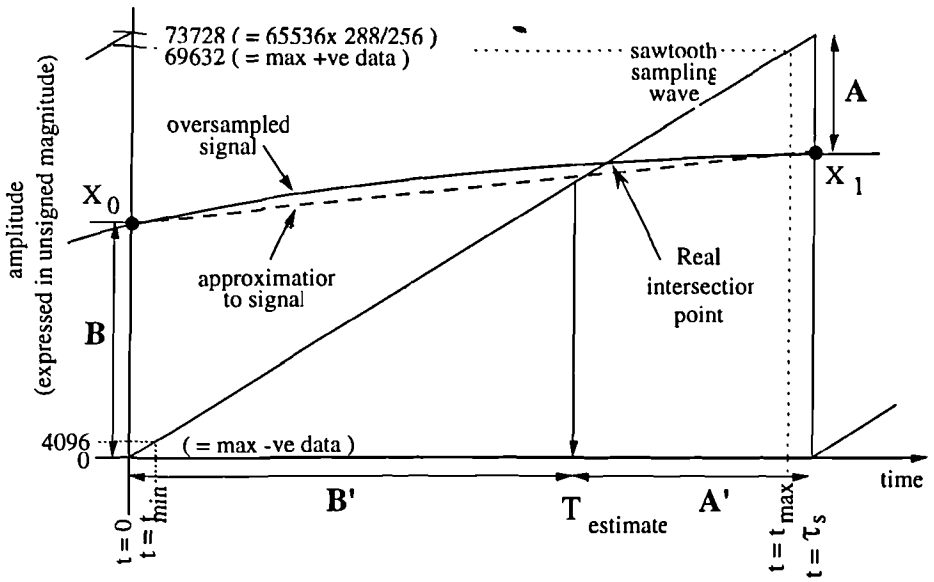


Figure 5.2.3.b : Calculation of The Weighted Average Value in 88% Modulated TEPWM

Taking a sixteen bit two's complement input, the data range is known to occupy -32768 to +32767 which gives a range of 65536 values. The signal is known to have no DC offset, and centre value zero, so the used range is actually -32767 to +32767. The output is to be modulated to 256/288 % modulation depth so the input should occupy ≈88% of the amplitude range, therefore the unsigned amplitude range will be 0 to 73727 (giving 288/256*65536 = 73728 levels). To re-zero the input in the centre of the new scale system, 73728/2=36864 must be added to each data value.

Taking X_0 and X_1 in the new scaling system, $\mathbf{A} = 73728 - X_1$, $\mathbf{B} = X_0$. Using an output scale system the same as the input allows a sampling sawtooth gradient of one; the time at which the pulse should end, $T_{estimate}$, will now take on the same value as the amplitude of the straight line approximation to the signal at the point at which the pulse ends. By similar triangles,

$$\frac{A}{B} = \frac{A'}{B'} \Rightarrow B' = \frac{A'.B}{A}$$

Equation 5.2.3.a

from the output scale, $A' = 73728 - B'$ Equation 5.2.3.b

so,

$$B' = \frac{(73728 - B').B}{A} \Rightarrow \frac{A}{B} = \frac{73728 - B'}{B'} = \frac{73728}{B'} - 1$$

$$\Rightarrow B' = \frac{73728 . B}{A + B}$$

$$\Rightarrow T_{estimate} = \frac{73728 . X_0}{73728 + (X_0 - X_1)} = \frac{X_0}{1 + \frac{X_0 - X_1}{73728}}$$

Equation 5.2.3.c

These parameters have been used in a programme running on a 96002 DSP IC with a version of equation 5.2.3.c . For this the algorithm is modified to avoid the division operation by reordering the denominator as ' 1 - Δ ' :

$$T_{estimate} = \frac{X_0}{1 - \Delta} \quad \text{where,} \quad \Delta = \frac{X_1 - X_0}{73728}$$

Equation 5.2.3.d

hence,

$$T_{estimate} \approx X_0 \cdot (1 + \sum_{R=1}^{N} \Delta^R)$$

Equation 5.2.3.e

By choosing 'N', the number of iterations of the summation such that Δ^{N+1} is smaller than the input quantisation level, and by appropriate nesting of the evaluation of it, a highly efficient pre-compensation algorithm can be written. The core of the algorithm is shown in block form below :

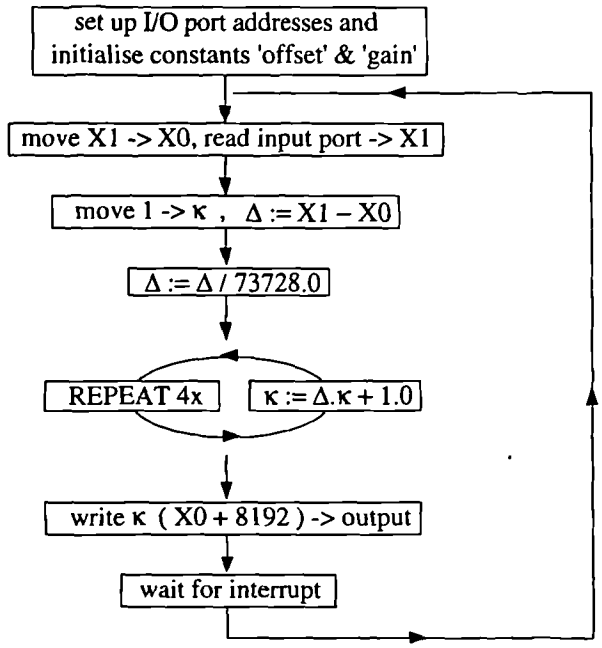


Figure 5.2.3.c : Pre-compensation Algorithm for Weighted Average PWM

This pre-compensation is particularly useful since it is good enough to enhance simple trailing edged modulators to similar performance to the two sample consecutive modulation type without the need for additional interpolation or the higher gain noise shaping associated with losing one bit in the data wordlength for the 2SCPWM case. The pre-compensated data can also be used as a better first guess to initialise higher order pre-compensation techniques (sections 5.2.5, 5.2.6 & 5.2.7).

To predict the spectral performance, double Fourier analysis can be used as in chapter two; as the pulse end time is no longer a simple trigonometric one, nor a simple translation of it, the Bessel function identity can no longer be applied. Alternative numerical evaluation can be used to calculate the final value of the distortion levels after simplification of the Fourier coefficients. This simplification is presented in appendix A8, but the complexity of this analysis strongly points towards the use of computer simulation for evaluation of the output performance.

The distortion performance of WAPWM as a function of modulation depth for the dominant (second) harmonic term and as a function of F_v/F_c is shown below (cf. figures 2.4.2.b, 2.4.4.a).

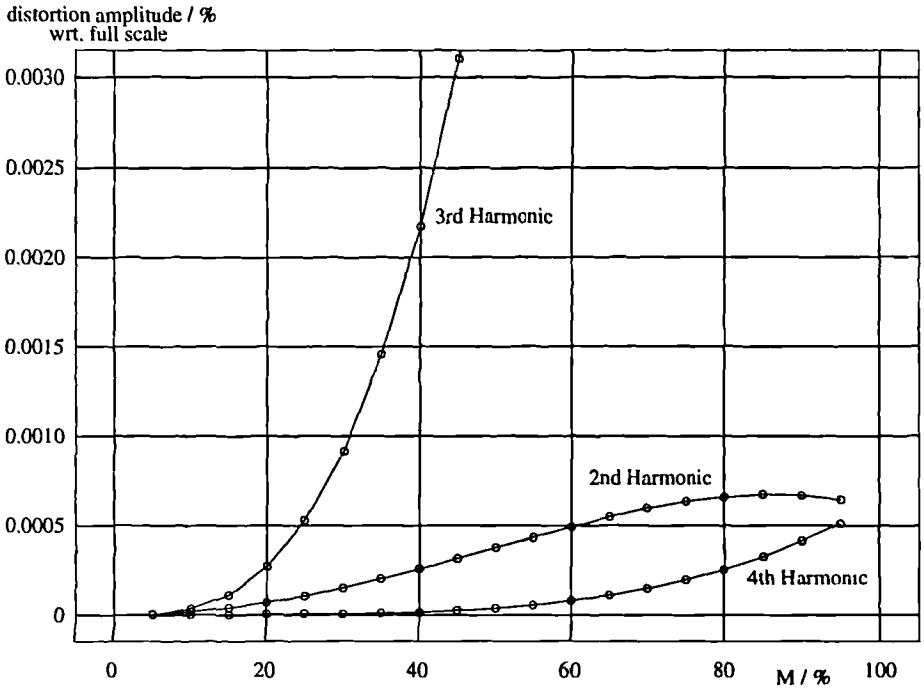


Figure 5.2.3.d : Dominant Distortion Level Plotted as a Function of Modulation Depth, "M".

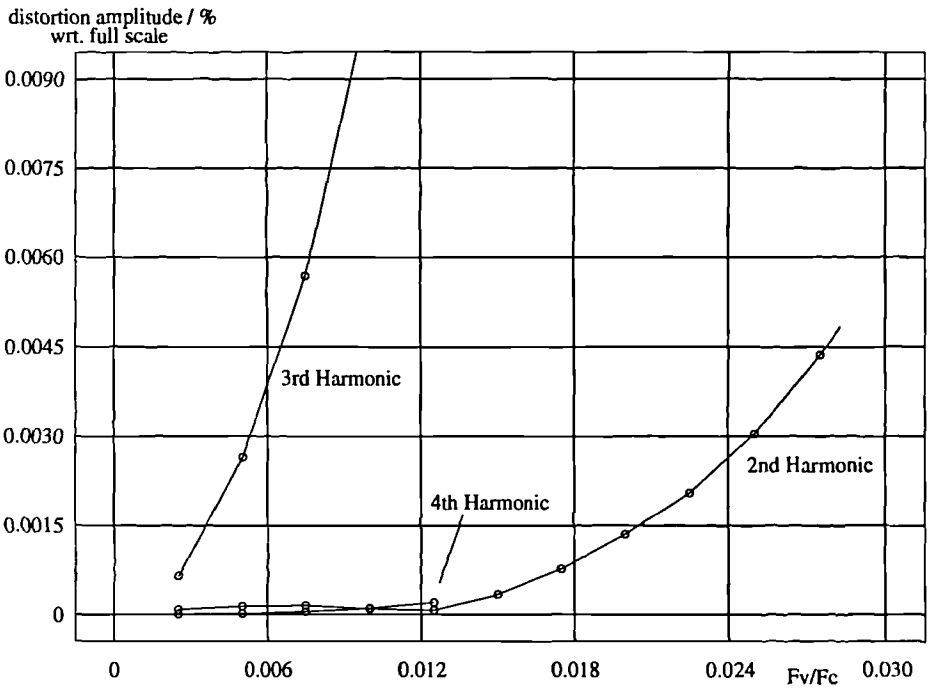


Figure 5.2.3.e : Dominant Distortion Level Plotted as a Function of Frequency Ratio, F_v/F_c.

As figure 5.2.3.d shows, the harmonic distortion after weighted averaging is better than with uniformly sampled TEPWM but is still large for reasonable pulse repetition rates. For greater accuracy in the approximation of naturally sampled PWM the signal must be known in more detail and one example in the literature 'LPWM', uses oversampling to realise further linearisation without increasing ω_c/ω_v [PED94]. Further interpolation within the pre-compensation permits closer straight line approximation of the signal over a zoomed-in subsection of the available period at a cost of doing the sample rate increase and a search to find in which sub-interval the intersection lies. Unfortunately the computational effort of this scheme is high, and the distortion reduction is not large without significant additional oversampling (typically a further 5x) so this technique will not been considered further.

5.2.4 : Pre-compensation Algorithms : Enhanced Sampling.

When oversampling has to be severely restricted, even lower than the minimum calculated in section 5.1, both sideband terms and harmonic terms contribute to the audio band distortion in the uniformly sampled PWM output spectrum. Naturally sampled modulation is known to have worse sideband performance than uniform, and uniformly sampled modulation is known to have worse harmonic performance than natural. A minimum in the total distortion can be found by taking a combination of the two sampled values which balances the level of the distortion from the two sources [MAL67] although the minimum harmonic distortion can never be as good as that provided by an oversampled, naturally sampled system.

It has been suggested that using the weighted average of samples as an approximation to naturally sampled values, and balancing these with uniformly sampled values can be modulated to produce an output spectrum with lower THD than uniform alone [LEI90a, LEI90b, MEL91]. Using the weighted average of samples instead of the true naturally sampled values leads to a small difference between the theoretical output spectrum and the measured output spectrum. Thus the THD at the theoretical balance (based on natural and uniform) is slightly different from the empirically measured balance (resulting from weighted average and uniform). However, there still exists a minimum in the THD at which the distortion is lower than that from a uniform modulator or weighted average pre-compensated modulator alone if : $\omega_c > 3\omega_{v(max.)}$ [MAL67].

The theoretical output spectrum of enhanced sampling is a generalisation of the uniform and natural sampling spectra as presented in chapter two. Taking a two sided case as an example [MEL91],

$$F(t)=\sum_{n=1}^{\infty}\frac{2HJ_n(Mn\,\pi\alpha\varepsilon/2)}{(n\pi\alpha\varepsilon)}\,Sin(n\pi/2).Cos(n\omega_v t-n\pi\alpha\varepsilon/2)$$
$$+\sum_{m=1}^{\infty}\sum_{n=-\infty}^{\infty}\frac{2HJ_n(M\pi(m+n\alpha\varepsilon/2))}{\pi(m+n\alpha\varepsilon)}\,Sin((m+n)\pi/2).Cos(n(\omega_v t)+m\omega_c t-n\pi\alpha\varepsilon/2)$$

Equation 5.2.4.a

where the variables are as used in chapter two except :

$\alpha=\frac{\omega_v}{\omega_c}$, ε = the proportion of uniform content
ie. $(1-\varepsilon)$ is the proportion of natural

and H = the height of the output pulses

For the case of ε=0, equation 5.2.4.a condenses to the spectrum for naturally sampled modulation, and for ε=1, uniformly sampled modulation. For intermediate values of ε there is a tendency for harmonic distortion to increase with ε and sideband levels to reduce with ε. The nature of the distortion can still be seen to be reduce with either smaller modulation index or larger oversampling ratio (smaller α).

The total harmonic distortion arising in the audio band varies with ε and has been evaluated by simulation [PAU92a] and a minimum occurs at ε=0.36875 as shown below (figure 5.2.4.a). This minimum has been found to occur at approximately constant ε across a wide range of modulation depth and signal-carrier frequency ratio. It is predominantly odd order harmonics that are reduced, so this kind of pre-compensation is most suited for use with 2SCPWM where even harmonics are not present (when k=0.5, cf. section 2.3.6).

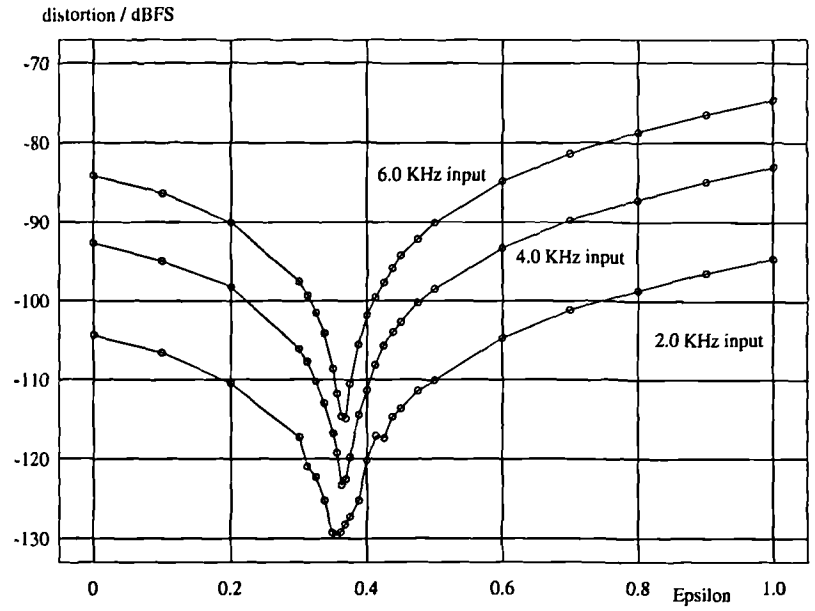


Figure 5.2.4.a : Output THD vs. 'ε' For Input Tones at Three Frequencies Using Enhanced Sampling.

Deriving the enhanced sample values is very similar to the processing for deriving the weighted average pre-compensated values except an additional term arises from adding in the required proportion of the uniformly sampled data. Figure 5.2.4.b below will be used as reference to explain the processing required.

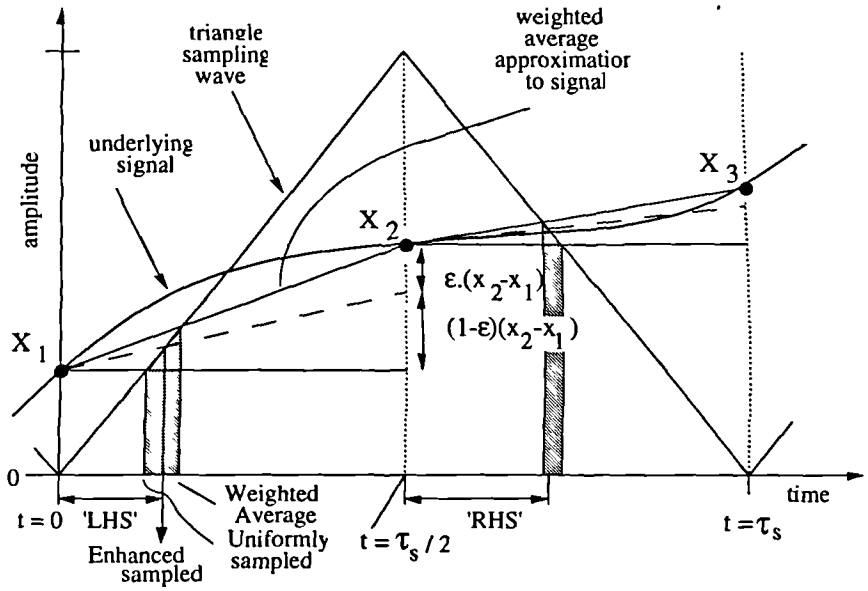


Figure 5.2.4.b : Deriving The Enhanced Sampled Values

Since the enhanced sampling process uses weighted average estimates of the natural sampling instants, division is required which may be simplified as in the pure weighted average case. The desired values of the the leading and trailing edges can easily be derived as before by similar triangles, and specified from the previous (left) regular sampling instant these are :

$$LHS = \frac{1 + X_1}{2 - (1 - \varepsilon).(X_2 - X_1)}$$

Equation 5.2.4.b

Similarly,

$$RHS = \frac{1 - X_2}{2 - (1 - \varepsilon).(X_2 - X_3)}$$

Equation 5.2.4.c

Starting with X_1 , the time of the left hand side of the pulse can be found by iterating with,

$$LHS_{n+1} = \left\{ \frac{(1 - \varepsilon).(X_2 - X_1)}{2} \right\} . LHS_n + \frac{1 + X_1}{2}$$

Equation 5.2.4.d

Similarly, starting with X_2 , for the right hand pulse edge time,

$$RHS_{n+1} = \left\{ \frac{(1 - \varepsilon).(X_2 - X_3)}{2} \right\} . RHS_n + \frac{1 - X_2}{2}$$

Equation 5.2.4.e

It is reported that five iterations of equations 5.2.4.d,e are required for convergence on the values produced by equations 5.2.4.b,c [LEI90a] but in experiments with sixteen bit data, and eight times oversampling, four iterations were found to be more than sufficient. This is most probably because the accuracy of the truncated approximate division improves dramatically as the dividend approaches one and for oversampled systems adjacent regularly sampled data are similar valued.

A particular example † from the literature [LEI90b] demonstrates the linearity improvements that can be offered by this system but since that system does not apply noise shaping, high resolution spectra are not available; noise-shaper / PWM interaction effects are not reported but have been found to be small in simulations using higher oversampling ratios (noise floor ≤ 120 dBFS).

It is interesting to note that in a system with two sources of error which vary in a complementary way, the minimum overall error occurs where the two sources of error are of equal magnitude. While this is obvious for orthogonal errors and linear systems, the harmonics and sidebands are not expected to be orthogonal and the distortion is known to arise from a system which is not 'linear, time-invariant'.

† For a system using enhanced sampling at 44.1 KHz (No oversampling) with ε=62, 2SCPWM modulated to 90% by a 6 KHz tone, the harmonic level is -53 dB at 18 KHz and the fourth lower sideband term is -54 dB at 20.1 KHz. Using a uniformly sampled signal alone the dominant distortion tone is the 3 rd. harmonic at -42 dB and using weighted average pre-compensated 2SCPWM the dominant distortion tone is the 4 th lower sideband of the carrier fundamental at -38 dB. Thus the enhanced sampling technique is said to have effected a 13 dB improvement in linearity.

5.2.5 : Pre-compensation Algorithms :
Pseudo-Natural PWM Using a Binary Search Technique

To get a more accurate representation of naturally sampled PWM than can be provided by weighted averaging, a more accurate version of the input signal is required. For test purposes this can be provided because the signal generator can be controlled directly, and the signal can be evaluated at any point in time. For a digital PWM which simply counts the width of pulses synchronously with a master clock, the required pulse widths (ie. durations in time) have to be found from the input signal (regularly sampled amplitudes). An algorithm for doing this can be derived from the analogue technique of comparison with a sawtooth. Where the signal is being generated artificially (eg. in simulation), the naturally sampled pulse durations can be found using a binary search.

A programme was written in PASCAL to do just this - 'natsigen.pas' - and is listed in appendix A1. It operates by generating regularly sampled data, and then splitting the current sample period in half. At this half way time, a new sample is generated and tested to compare with a sawtooth function of full amplitude. The right or left half is subsequently split and the operation repeated, depending on whether the last sample was greater or less than the sawtooth. For 'n' bit data, n+1 divisions are required (at most), so after this many the time found is dithered, rounded, and output as a sufficiently close approximation to the intersection point for natural sampling. This programme also adds appropriate headers for analysis by Audio Precision's System One (Dual Domain) but formatting for this system limits the signal resolution to 24 bit (maximum). Signal frequencies for tests were specifically chosen to avoid integer ratio with the sampling ratio so that a realistic quantisation noise floor could be seen despite the signal being a pure tone (which would otherwise have harmonically related quantisation noise). Spectra of the two signals it produces (the regularly sampled and the binary searched natural approximation) are shown below.

Floating point NTF noise shaping, trailing edged PWM and output decimation for analysis were also written to test the simulation of natural sampling. Programmes for these can also be found in appendix A1 and A2.

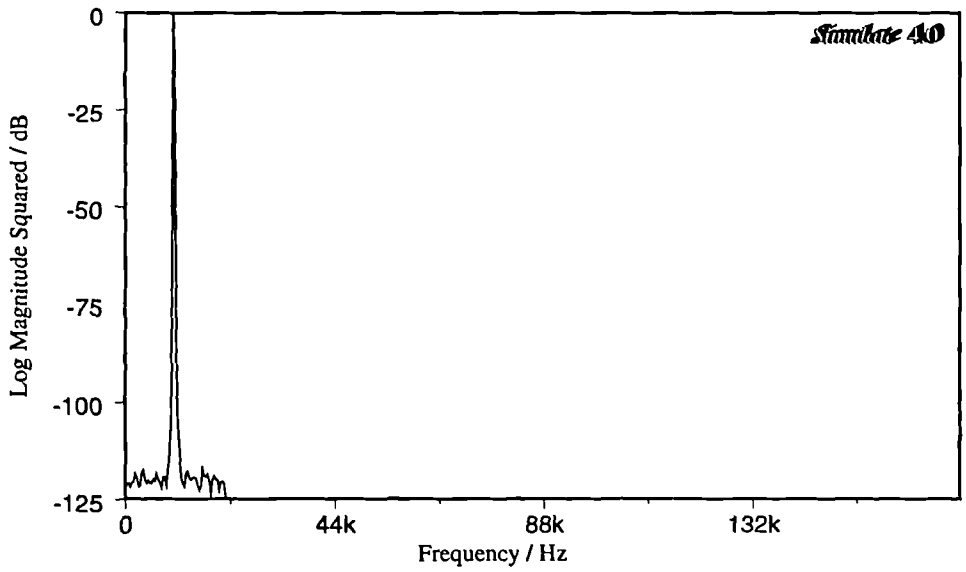


Figure 5.2.5.a : Regularly Sampled Spectrum of an Interpolated 16 bit Sinusoid

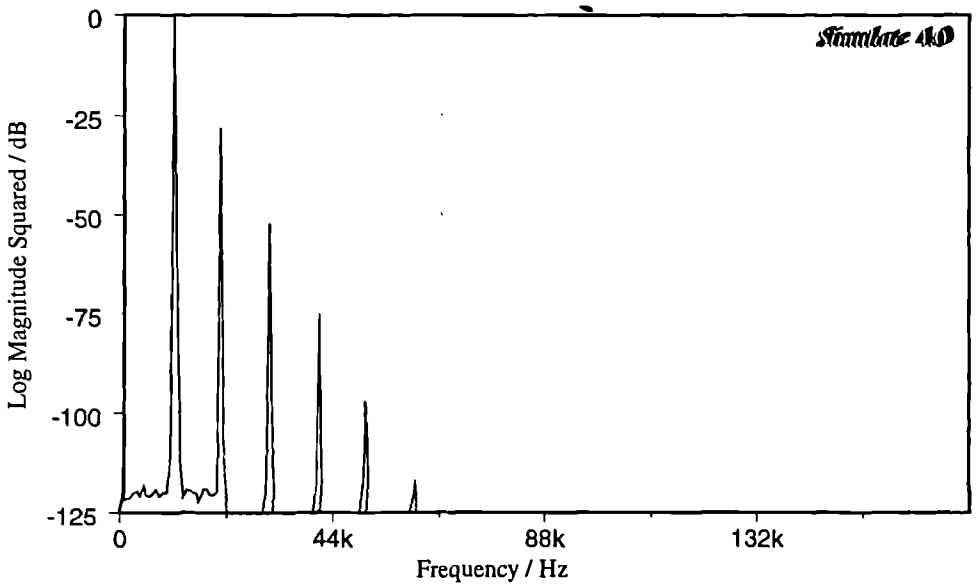


Figure 5.2.5.b : Naturally Sampled Spectrum of an Interpolated 16 bit Sinusoid

From the above figures it is clear that the naturally sampled signal has been distorted in a similar fashion to that which uniformly sampled TEPWM produces. This is encouraging since oversampled PWM is known to be close to linear, in so far as errors are of the order of one part in a thousand or less, so pre-distortion is expected to be an anti-phase version of the distortion with which is has to cancel. In short, the magnitude spectra of natural sampling and uniformly sampled TEPWM should be very similar. The required pre-compensation cannot be predicted perfectly in this way since the transfer function of the PWM is still not LTI even if nearly so (THD ≈ 0.06 % for a -15 dBFS tone @ 1 kHz in an 8x oversampled TEPWM).

Having noise shaped these signals to eight bits using a fourth order sinusoidal noise shaper, the outputs can be pulse width modulated and decimated to examine the baseband. Shown below are the output spectra for the two input signal spectra presented in figures 5.2.5.a and 5.2.5.b :

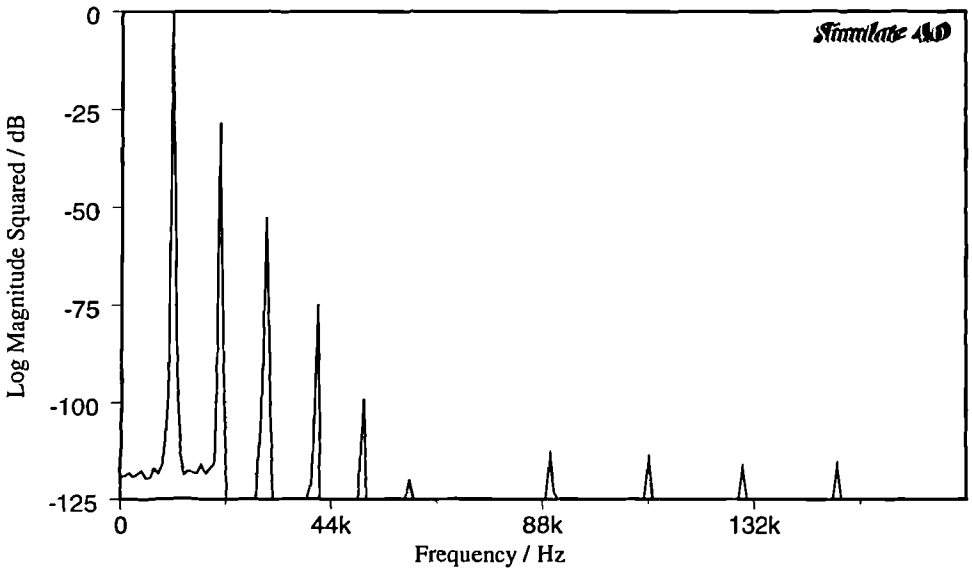


Figure 5.2.5.c : Uniformly Sampled TEPWM Spectral Output

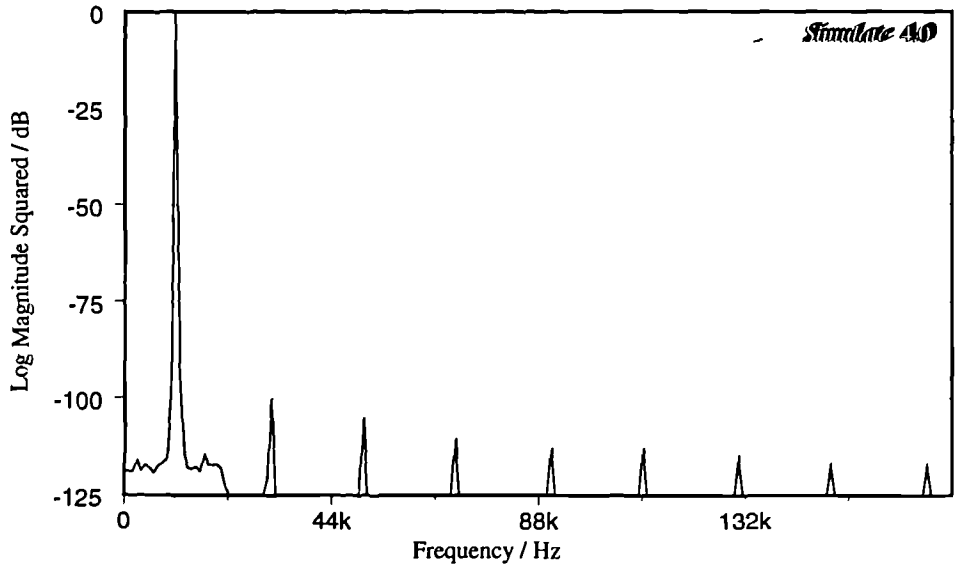


Figure 5.2.5.d : Pseudo Naturally Sampled TEPWM Spectral Output.
(nb. : the 'distorted' input has led to this undistorted output).

It is important that the scaling and offsets used in the pre-compensation are appropriate for the PWM which follows it. Since the signal after the pre-compensation and before the PWM has been non-linearly compensated, linear operations on it (such as the addition of an offset) make the pre-compensation inappropriate. The addition of shaped requantisation noise is in fact questionable since this would be expected to intermodulate with the pre-compensated signal. Keeping the requantisation noise amplitude small, and knowing that the oversampled PWM non-linearity is small seems to enable this operation to be completed without dramatic effect on the output.

From preliminary experiments such as shown above, this approach can be shown useful for linearising PWM DACs at up to 'CD' quality. For real time signals the processing must be capable of finding the intersection times more quickly than binary searching can provide and the signal must either be known as a function, or closely approximated by a polynomial, as discussed in the previous section. Speed, flexibility and accuracy of the intersection finding algorithm are the subjects of the next section.

5.2.6 : Pre-compensation Algorithms : Pseudo Natural PWM using Lagrangian Interpolation & Newton-Raphson Root Approximation.

By posing the intersection finding problem as that of finding the root of another function, the difference between the signal and the comparison sawtooth, the binary search can be replaced by faster root finding techniques. For its speed and simplicity, Newton-Raphson iteration is proposed to find the root of this difference function. For this to be applied the function must be monotonic in the required interval, have a non-zero gradient at all evaluated points and have no discontinuities. For fast convergence on the root a good initial guess is required with which to apply the iterative formula :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Equation 5.2.6.a

Since an interpolated signal cannot vary as fast as the rate of change of the sawtooth, the difference function must be monotonic, and have a non-zero gradient. The signal has also been low pass filtered in the process of interpolation so discontinuities cannot exist within the interval.

Operating the iterative formula requires approximation of the difference function, $y(x)$, rather than approximation of the signal, $S(x)$, but polynomial approximation can still be easily applied if the sawtooth is 'unwrapped' over the interval of interest to be a straight line passing through zero at 'X_1'. Thus the four signals of interest are defined as :

The input :	$S(t)$, sampled at four equi-spaced points in time $S_i(x)$,	
the sawtooth :	$\Delta(t)$, sampled at four equi-spaced points in time $\Delta_i(x)$,	
the difference :	$y(t)$, sampled at four equi-spaced points in time $y_i(x)$, and	
the timebase :	$X(t)$, with four sampling instants 'R' seconds apart, X_1, X_2, X_3, X_4	
such that :	$S_1(x) = S(t) \quad \mid x = X_1$,	
	$\Delta_1(x) = \Delta(t) \quad \mid x = X_1$,	
and :	$y_1(x) = y(t) \quad \mid x = X_1$.	Equation (set) 5.2.6.b

This new signal, $\Delta(t)$, is a triangle wave that overlaps adjacent evaluation periods each time a new sample is calculated. This does add the constraint that approximation polynomial orders higher than the interpolation factor should not be used to ensure monotonicity in the difference function for the application of the Newton-Raphson formula. The new root finding problem is show below :

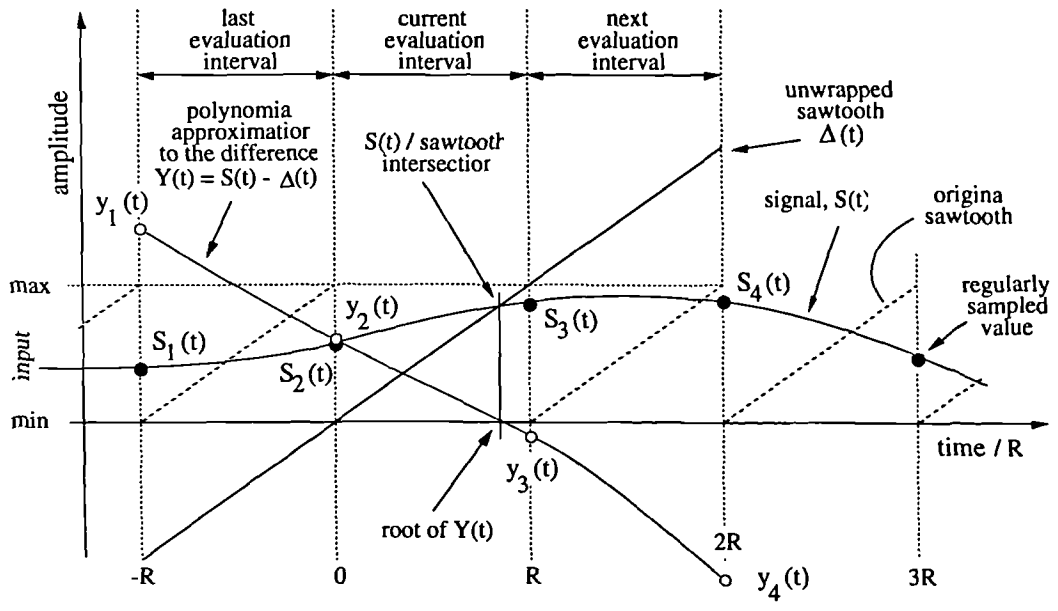


Figure 5.2.6.a : New Signal Definitions to Convert Intersection Search to a Root Finding Problem.

The technique of Lagrange polynomial approximation for four support abscissae was attempted. A third order polynomial is constructed knowing that the approximate polynomial must pass through the known regularly sampled points. By expressing the polynomial as the sum of four products of Lagrange Polynomials and the support abscissae :

$$P_4(x) = \sum_{i=1}^{4} \{ y_i . N_i(x) \}$$

Equation 5.2.6.c

where,

$$N_i(x) = \prod_{k=1, k\neq i}^{4} \frac{(x - x_k)}{(x_i - x_k)}$$

Equation 5.2.6.d

and hence, for the case of x_1, x_2, x_3, x_4 equi-spaced by 'R' in ascending numeric order,

$$P_4(x) = \frac{1}{6.R^3} \begin{bmatrix} y_4.(x-x_1).(x-x_2).(x-x_3) \\ -3.y_3.(x-x_1).(x-x_2).(x-x_4) \\ +3.y_2.(x-x_1).(x-x_3).(x-x_4) \\ -y_1.(x-x_2).(x-x_3).(x-x_4) \end{bmatrix}$$

Equation 5.2.6.e

where y_1, y_2, y_3, y_4 are the samples of the difference signal, $y(x)$, = $S(x) - \Delta(x)$.

To convert the timebase 'x' to clock cycles or seconds, the scale factor 'R' has to be defined which will depend on the modulation depth required and the number of bits resolution available at the input of the noise shaper. In general, the algebra is simplest if R is set equal to one and re-scaled afterwards. The polynomial can be rewritten :

$$P_4(x) = \frac{1}{6.R^3} \begin{bmatrix} y_4 - 3y_3 + 3y_2 - y_1 \\ 3y_3 - 6y_2 + 3y_1 \\ -y_4 + 6y_3 - 3y_2 - 2y_1 \\ + 6y_2 \end{bmatrix} . \begin{bmatrix} x^3, Rx^2, R^2x, R^3 \end{bmatrix}$$

Equation 5.2.6.f

from which the derivative can easily be calculated :

$$P_4'(x) = \frac{1}{6.R^3} \begin{bmatrix} 3y_4 - 9y_3 + 9y_2 - 3y_1 \\ 6y_3 - 12y_2 + 6y_1 \\ -y_4 + 6y_3 - 3y_2 - 2y_1 \end{bmatrix} . \begin{bmatrix} x^2, Rx, R^2 \end{bmatrix}$$

Equation 5.2.6.g

Using the above two equations substituted in equation 5.2.6.a, arbitrary input sampled amplitudes can be pre-compensated to approximate naturally sampled pulse durations so that after noise shaping, modulation and decimation, a linear version of the input data is created. These equations were programmed into 'lanr.pas' (see appendix A.1.3) to take sixteen bit, regularly sampled data and output 16 bit pre-compensated data. A flow diagram of this is shown below :

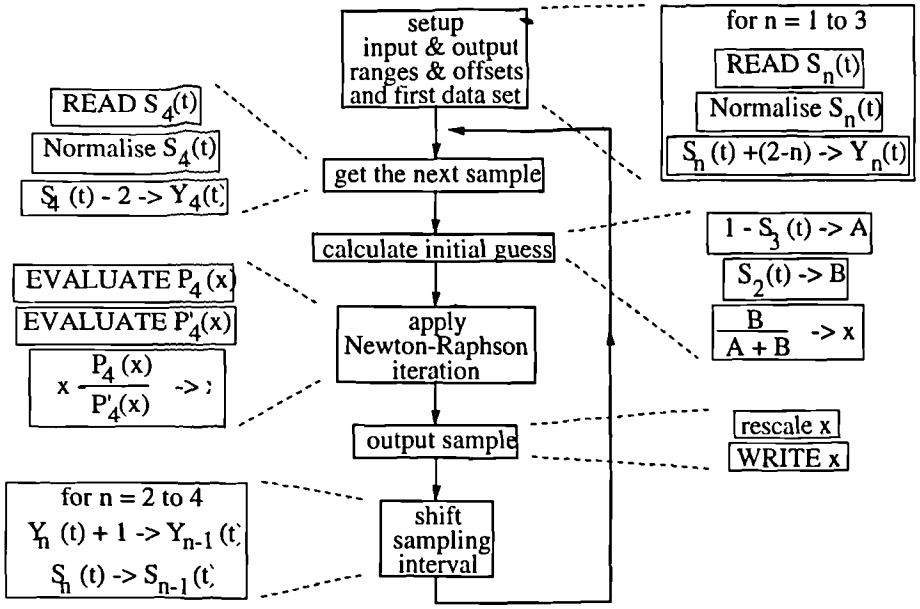


Figure 5.2.6.b : Flow Diagram of Pre-compensator 'lanr.pas'.

It was found by experiment that weighted average initial guess values allowed a good enough first estimate of the naturally sampled data to apply only one iteration of equation 5.2.6.a for almost totally linear results in a sixteen bit system. Also, more effective compensation could be performed by using either higher oversampling ratios, lower amplitude signals, higher order polynomial approximation, closer initial guesses or more iterations of the Newton-Raphson formula.

Unfortunately the above procedure, even in its most basic form is extremely difficult to operate in real time on a DSP IC. More elegant solutions to finding the sawtooth intersections have been investigated [GOL92] and are discussed in the next section.

5.2.7 : Pre-compensation Algorithms : Pseudo Natural PWM using Newtonian Polynomial Interpolation, a Difference Table With Retained Intermediate Terms & Newton-Raphson Root Approximation.

Alternative calculation of the approximation polynomial can be used, knowing that the third order approximate polynomial derived from the four support abscissae is unique for those abscissae [ATK89]. This has been done for the third and fifth order cases with error analysis in [GOL92]. The third order case will be presented below for comparison with the techniques used in the last section. Evaluating the 'Newton polynomial' form (as shown in equation 5.2.7.a) is particularly easy [SCH89] since the coefficients of the polynomial when written in this form (the 'Newton coefficients') can be derived recursively, using a divided difference table.

$$P_n(x) = C_0 + C_1.(X-X_0) + C_2.(X-X_0).(X-X_1) + \ldots + C_n.(X-X_0).(X-X_1)\cdots(X-X_n)$$

Equation 5.2.7.a

When the support abscissae are equi-spaced the divided difference table degenerates into a simple difference table making it particularly efficient at finding the Newton coefficients. In this case

the samples from y(x) are used as the start column in a four column difference table, reading off the Newton coefficients from the leading diagonal:

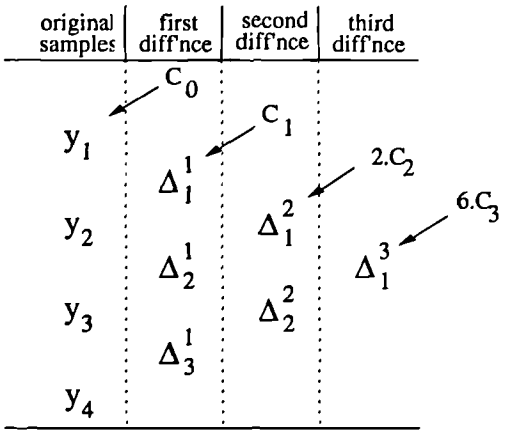


Figure 5.2.7.a : Four Column Difference Table
For Evaluating The Unique Third Order Polynomial's Coefficients in The Newton Form.

When evaluating the Newton polynomial at the initial point used in equation 5.2.7.a, the terms can be calculated efficiently by nesting the multiplications as shown below, making this evaluation well suited to DSP architectures which allow concurrent multiply and add [GOL92].

$$P_3(x) = c_0 + (x-x_0)\left[c_1 + (x-x_1)\left[c_2 + (x-x_2)\left[c_3\right]\right]\right]$$

Equation 5.2.7.b

By storing each of the bracketed intermediate terms b_3, b_2 and b_1,

$$b_3 = \left[c_3\right]$$
$$b_2 = \left[c_2 + (x-x_2)\left[b_3\right]\right]$$
$$b_1 = \left[c_1 + (x-x_1)\left[b_2\right]\right]$$

Equation 5.2.7.c

the derivative can be simplified when evaluated at the same point, x, this can be expressed as [GOL92]:

$$P'_3(x) = b_1 + (x-x_0)\left[b_2 + (x-x_1).b_3\right]$$

Equation 5.2.7.d

Although the third order example has been worked through here, the same approach can be applied to arbitrary order polynomial approximation and speeding up the pre-compensation becomes increasingly important for more accurate pre-compensation since this requires a higher order polynomial. If a dual data bus processor can be used for this procedure, a very streamlined algorithm can be developed since the intermediate quantities (the 'b's) can be concurrently moved to memory while the polynomial is being evaluated. This still requires considerable computation in a short time which is shortened by oversampling (although this makes the approximation more accurate).

A further reduction in computational complexity can be achieved by using all the intervals of the polynomial. While this can be used in the third order case without degrading performance, approximation error increases so much in the outer intervals for the fifth order case that only three of the five intervals can be used from one set of support abscissae. Further small differences in the

computational complexity can be traded off against accuracy by operating with an approximation to S(x) or y(x) and by using pre-scaled data.

Simple third order approximation of the signal suffices to eliminate distortion for most signals but for full amplitude tones especially at high frequencies, intermodulation distortion does appear at low levels (-117 dBFS). The third and fifth order cases have been evaluated [GOL92] to compare the computational complexity of the above approach and these are shown below.

Processing	Third Order	Fifth Order
Calculate a first guess, x	3 adds, 2 multiplies, 1 divide	3 adds, 2 multiplies, 1 divide
Evaluate Difference Table	6/3 adds	14/3 adds
Calculate c_n	2/3 multiplies	4/3 multiplies
Calculate $(x - x_n)s$	3 adds	5 adds
Calculate b_n	3 adds, 3 multiplies	5 adds, 5 multiplies
Evaluate the polynomial	1 add	1 add
Evaluate the derivative	3 adds, 2 multiplies	5 adds, 4 multiplies
Use Newton Raphson once	1 add, 1 multiply, 1 divide	1 add, 1 multiply, 1 divide
TOTAL	16 adds, $8\frac{2}{3}$ multiplies, 2 divides	$24\frac{2}{3}$ adds, $13\frac{2}{3}$ multiplies, 2 divides

Figure 5.2.7.b : A Table Comparing Polynomial Approximation Complexity

(nb. When measured in this way, the process outlined in the last section and used in simulation in programme 'lanr.pas' (3 rd. order, weighted average guess, 3 intersects per polynomial) would require more multiplication : $12^2/_3$ adds, 13 multiplies , 2 divides. With appropriate use of multiply and add instructions on a DSP this may not be much worse; this will be 'target' processor dependent).

5.2.8 : Pre-compensation Algorithms : Pseudo Natural PWM Using Inverse Interpolation

Although the last two sections have concentrated on the potential of using forward interpolating polynomials, there exists an alternative approach, namely of using the inverse interpolating polynomial [GOL92].

In this approach the support abscissae are used as unevenly spaced ordinates, and the regularly spaced ordinates are used as support abscissae. The forward polynomial is required to be monotonic in the interval to ensure that an inverse exists but this is always satisfied if the difference signal y(x) is used instead of the signal itself S(x) (as in the Newton-Raphson approach). To find the intersection point the inverse interpolating polynomial has to be evaluated once, at the value $y(x) = 0$, but finding this polynomial may not be easy. Fortunately, evaluating the polynomial does not require Newton-Raphson iteration with its associated complexity of finding an initial guess, evaluating the derivative and division in the iterative formula.

The inverse interpolating polynomial can be derived using the tabular technique of the previous section, but, the divided difference table cannot be decomposed into the difference table because the new ordinates (the old support abscissae) are unevenly spaced. This leads to a dramatic increase in the computational complexity making this an unattractive technique for current DSP capabilities, so it will not be considered further.

5.3 : Open Loop PWM DACs using Non-Stationary Pre-compensation.

5.3.1 : System Identification.

Pseudo natural pre-compensation is simply an algorithm for emulating natural sampling which is an example of a non-uniform sampling process. Such a pre-compensation can be viewed as a time variant linear process rather than a true non-linear process since the pulse area (ie. output signal energy) is linearly linked to the input signal amplitude. By elimination this implies that it is the time varying sampling instant which leads to distortion in the output. From this it follows that time variant filtering should be the appropriate 'antidote' for PWM's distortion.

Within PWM, two processes can be seen (i) the representation of sampled input energy by the output, and (ii) the representation of the input samples' placement in time by the output. If the input samples are defined as a set, x(t), where t is the sampling time (which is quantised in a regularly sampled data stream), then PWM can be seen to maintain the first moment of x, (the integral of each PWM pulse is proportional to the integral of the PAM sample from which it came). However, the second moment of x is not maintained as the energy becomes spread in time away from its 'timing marker'. Furthermore, the output will become phase distorted in certain cases because pulses from larger amplitude samples will be centred later (or earlier) in time for certain modulation types.

It is interesting to note that where LEPWM and TEPWM with equivalent modulation parameters and combined to yield SYMPWM or 2SCPWM, the phase cancellation would be expected to cancel (and the distortion is better in these modulation types). Also, if LEPWM and TEPWM are combined in such a way that their modulation parameters are not the same (eg. 2SCPWM with $k \neq 0.5$), the distortion becomes considerably worse (phase cancellation of errors cannot occur in this case).

For a more linear performance from PWM, the signal must be compensated not to emulate natural sampling itself, but to counteract both the phase distortion and the spreading of energy implicit in using pulses to represent the input.

In a more general sense the noise shaper and PWM should be characterised as a non-linear system so that sideband and intermodulation distortion could be taken into account as well as the time variant distortion introduced by using uniform modulation. Volterra series are popular for system modelling of this kind but remain only approximate, requiring the truncation of an infinite series. The computational complexity of such an approach is enormous, so it is though unlikely that this approach will be able to contend with the already proven efficiency of the PNPWM technique. However, time variant filtering schemes have been devised, not by theoretically identifying the sources of error, but by compensating for the effects measured in the frequency domain from modulating two typical signals [HAW92]. This technique will be discussed in the next section.

5.3.2 : Model-Based Dynamic Linearisation.

By viewing each PWM pulse as a spectrally altered representation of the sampled value from which it was derived, a compensating filter from a bank of filters can be used to spectrally compensate for each pulse according to its value [HAW92]. While this kind of linearisation was first aimed at low oversampled systems the underlying approach of a time variant filter for pre-compensating PWM can be applied in many cases. The pre-compensating element can be placed in a number of places in the open loop structure of figure 5.1.1.a, but in essence its operation can be conceived as a replacement for the pre-compensating element as shown in figure 5.2.1.a .

Since the spectral contribution of each pulse will be spread by its pre-compensating filter, for an 'n'-tap FIR filter (assuming all the filters have the same odd valued length, n) the pre-compensation will contribute to (n-1)/2 adjacent samples both forward and backward in time. To make this system causal, a delay of at least 1+(n-1)/2 has to be included. Also, since each output sample now depends on its associated input sample and those adjacent to it, an overlap in the pre-compensating filters will occur. To properly account for this overlap, not only the 2^b filters for each input value has to be used but in fact $n!.2^b$ to cover all the input combinations over the length of the filters. In practise one signal-dependent, time-variant filter can be used to approximate all the filters, and polynomial approximations to the conversion functions between the local input signal sequence and the current coefficients can be used.

The specific published example [HAW92] uses n=5 so the structure of the time varying filter used as a pre-compensator is as shown in figure 5.3.2.a.

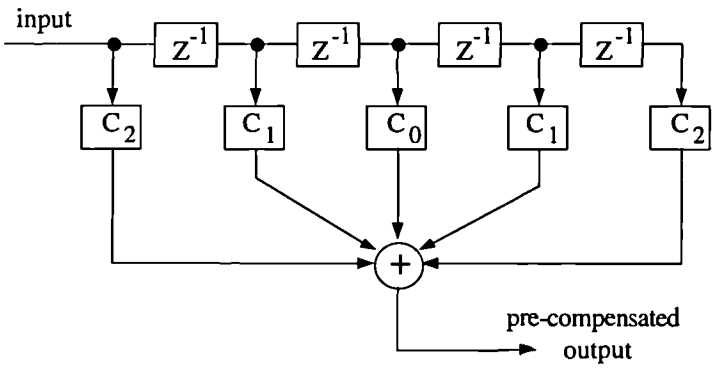


Figure 5.3.2.a : Time Varying FIR Filter For PWM Pre-compensation.

By attempting to use a symmetric PWM modulation type, the problem of phase distortion and parasitic AM->PM conversion can be avoided (this can stem from pre-compensation applied to signals as well as shaped requantisation noise - see section 4.5.5). Furthermore, because the modulation does not exhibit dynamic phase modulation which would need to be corrected for in the single sided case (larger amplitude...later pulse centre) a linear phase pre-compensation can be used, ie. symmetric FIR filtering. In the fifth order case, which will be used as a design example, this implies using a coefficient set $\{C_2, C_1, C_0, C_1, C_2\}$. Since the DC gain should be unity, C_0 can be derived from the other coefficients leaving in the general case (n-1)/2 free coefficients.

To design the coefficient sets for each filter (ie. for each input configuration) the Fourier transform of the filtered pulse width modulated signal should yield a constant spectrum which will be referred to as the target function. To simplify the steady state case, the target function, T(f), is chosen to be the spectrum of a 1:1 mark to space ratio square wave at the carrier frequency, normalised to be unity at DC; this imitates the PWM output for the no signal case:

$$T(f) = \frac{\mathrm{Sin}(0.5\pi f T_s)}{0.5\pi f T_s}$$

Equation 5.3.2.a

The transfer function of the pre-compensating filters, E_x , for each input, x, have the general form:

$$E_x = C_0 + 2C_1\,\mathrm{Cos}(2\pi f T_s) + 2C_2\,\mathrm{Cos}(4\pi f T_s)$$

Equation 5.3.2.b

and the uncompensated PWM output spectrum for a pulse of normalised width x ($x = 1 \Rightarrow \text{width} = T_s$)

$$F_x(f) = \frac{\mathrm{Sin}(X\pi f T_s)}{X\pi f T_s}$$

Equation 5.3.2.c

so to linearise the output, the product of Ex(f) and F(f) must equal T(f) over at least the audio band :

$$E_x(f).F_x(f) = T(f)\Big|_{0<f<f_a}$$

Equation 5.3.2.d

By observing that the values of C_1 and C_2 have similar curvature for low frequencies, their values can be approximated by solving simultaneous equations which exactly satisfy equation 5.3.2.d at two frequencies, the end of the audio band and half way through (11 & 22 KHz). The solutions provided from this yield close approximation over the band specified for x=0.5 since this was the declared target function (new target functions are declared for each input value, x). Better general values for C_1 and C_2 can be found by optimisation, minimising the error in 5.3.2.d over the entire band of interest rather than at two frequencies alone.

Three implementations of these results can be used; direct implementation of the solutions of the simultaneous equations (or optimisation), a ROM look-up table for some discrete values, or polynomial approximation to the variation in C_1 and C_2 as a function of the input value (higher order filters will require more polynomial approximations). The polynomial case is most useful in high accuracy systems since the ROM becomes unwieldy and it is computationally less intensive than the direct implementation approach.

Having derived the independent filters as a function of input amplitude, interaction between adjacent pulses' compensated spectrums needs to be taken into account in the filters. Since each filter would modify the adjacent sample amplitudes (both forward and backward in time) and those will in turn modify the current pulse's amplitude, the coefficients are modified in a recursive manner. Fortunately, $C_0 \gg C_1 \gg C_2$, so the propagation of this recursive interdependence only spans a few samples (typically 18 - depending on the system resolution) and only some four or five iterations of a modification algorithm are required to finalise the values used for $C_0(x)$, $C_1(x)$, and $C_2(x)$.

This pre-compensator, like most others presented in this chapter, produces an output which is highly resolved. This would normally imply that pre-compensation should be carried out for the signal

and the pre-compensated result should then be noise shaped to make the PWM realisable. It has been suggested [HAW92] that the above pre-compensation scheme can be used after noise shaping which implies attempted linearisation of the shaped requantisation noise. Since the pre-compensator was only designed to linearise up to the end of the audio band it is not understood how this may be the case, but there still remains the problem of the pre-compensator spoiling the wordlength reduction achieved by the noise shaper. In the literature, subsequent first order sinusoidal noise shaping (just before the PWM) is used to reduce the effects of this problem; higher order shaping is said to damage the linearisation requiring a linearity/resolution balance. Since the example presented only used a few input signal examples it is not clear whether first order sinusoidal noise shaping would be adequate for 'CD' quality although this approach clearly demonstrates the usefulness of time variant pre-compensation.

5.4 : Closed Loop PWM DACs Using Real Time Output Decimation.

5.4.1 : Feasibility Study.

In this section PWM based DACs using signal and error feedback and error feedforward will be looked at. Negative feedback is common in analogue amplifiers to reduce the effects of non-idealities in the open loop amplifier, and feedforward compensation is used in some power amplifiers (eg. the Quad 405, 'current dumping' amplifier). Theoretically, both feedforward and feedback signals are required for perfect linearity [WAL75, VAN80] since negative feedback alone is limited by stability problems. For a first look at closed loop PWM DAC architectures these will be considered one at a time. The architectures under consideration involve :

1) feedback of error or signal locally (as is the case for a noise shaper or ΣΔ modulator),
2) feedback of error or signal globally (ie. around more than just one system block), and
3) error feedforward using two PWM DACs or by comparison with a PAM based DAC.

A structure suitable for implementing (1) above is shown below. In this case the feedback of error has been shown in both cases (around the noise shaper, and around the PWM) [SAN91c].

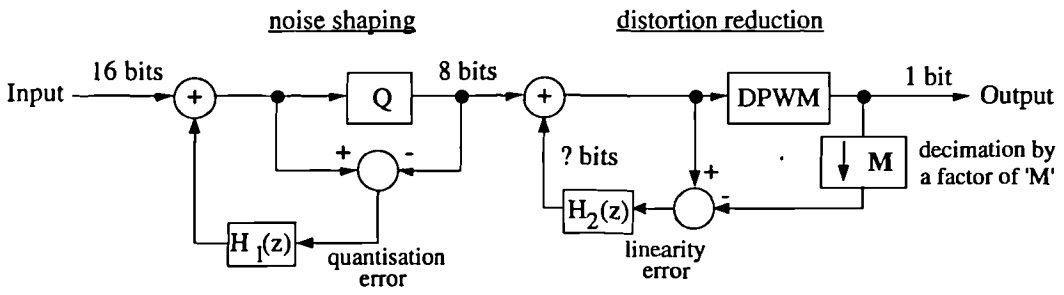


Figure 5.4.1.a : Local Error Feedback Architecture For a Closed Loop PWM DAC.

The structure shown above shows a standard noise shaper (which uses error feedback), followed by a digital PWM in a similar loop. It should be noted that as the output signal will be at a far higher sampling rate than the input to the PWM so the output must be decimated as indicated to evaluate the error from the PWM. Secondly, the pulse width modulator is digital so its input is of limited wordlength, ie. requantised. Although the signal processed by the PWM is of limited resolution, it is highly accurate so simple feedback as shown will have to support the full system resolution. This introduces problems since any signal added to the input of the PWM which is resolved more finely than the PWM can handle must either be quantised (and lost) or noise shaped. If noise shaping of this error signal has to be applied, a global feedback structure can be used and is simpler if as shown below :

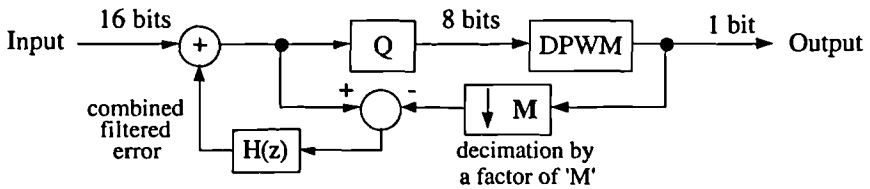


Figure 5.4.1.b : Closed Loop PWM DAC Architecture With 'Global' Error Feedback.

The above architecture still suffers from difficulty in evaluating the error since the decimation filtering introduces delay. To allow for this, an equivalent delay is required between the quantiser input and the feedback adder as shown below :

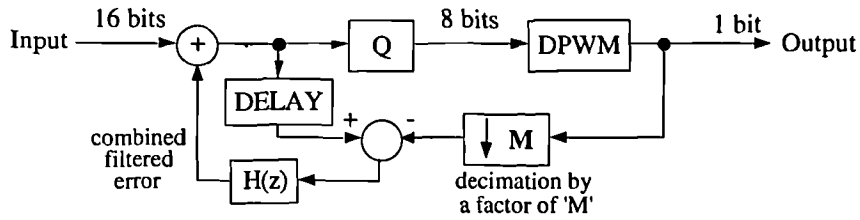


Figure 5.4.1.c : 'Global' Error Feedback Architecture With Processing Delay Correction.

This delay will be seen in the next section to be the limiting factor for the usefulness of such a system, because the stability of the loop becomes affected when it is large, restricting the bandwidth that can be linearised. Before developing this structure further, two feedforward structures will quickly be looked at since feedforward linearisation does not suffer the same stability constraints as feedback linearisation.

The first of the two feedforward schemes involves using two PWM based DACs, one to operate in an open loop fashion which can be measured, and a second which can be given a pre-compensated signal made up of the input minus the error measured from the first DAC. For this to operate properly, the processing delay of the first DAC must be taken into account as shown below :

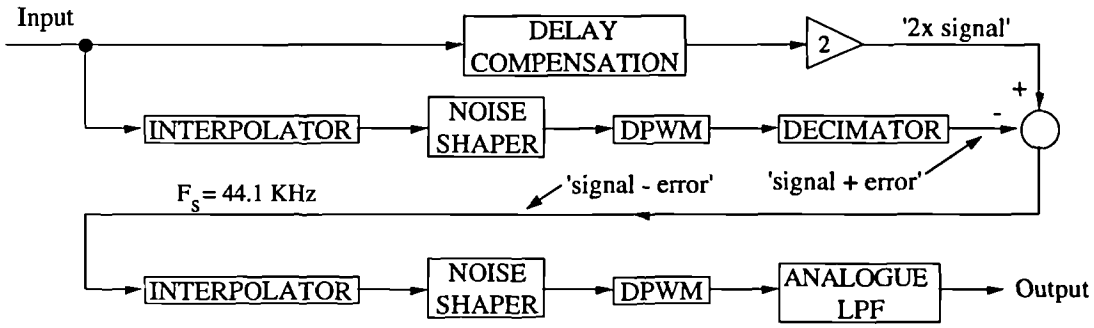


Figure 5.4.1.d : Feedforward Correction Using a Dual DAC System

It should be noted that such a structure takes advantage of purely digital processing right up to the output LPF. This theoretically enables high accuracy to be maintained in the pre-compensation scheme. Furthermore, since the output of the first PWM will include both harmonic and sideband distortion elements, this structure can compensate for both these sources of error.

However, this system cannot totally linearise the overall D/A conversion because it relies on superposition of the error on the signal to cancel the output error. As discussed earlier in this chapter, although PWM can be made linear to better than 0.1% by appropriate choice of modulation parameters, it is still not an LTI process and superposition is only approximately followed. Also, this system introduces considerable delay as a result of three blocks of sample rate change.

If large delay is unacceptable, some of it can be recovered by avoiding the decimation stage.

An architecture employing a combination of PWM and PAM DACs can achieve this, as shown below, but its performance will be limited by the analogue processing stages. The main benefit of such a scheme is the high efficiency output stage which this still makes use of, since the preceding signal processing (analogue or digital) can be performed at low level.

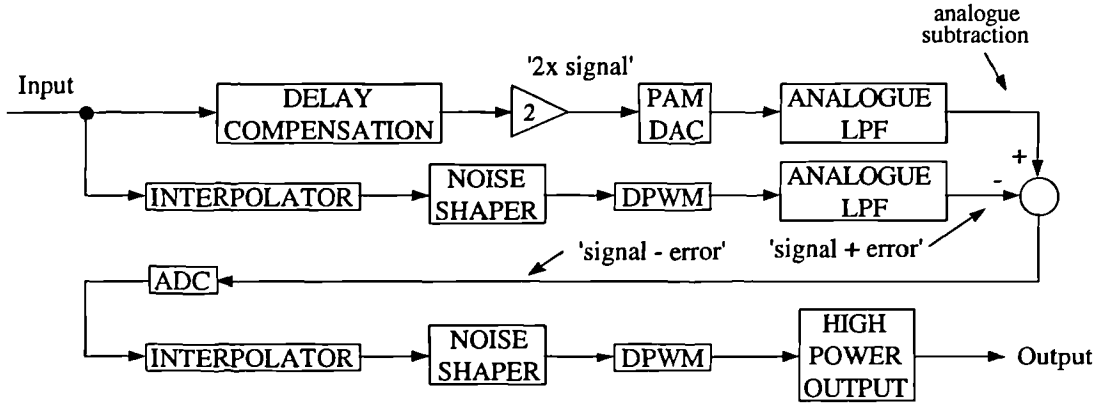


Figure 5.4.1.e : PCM Reference DAC, Linearising a PWM DAC.

To summarise, the best potential architecture of a closed loop PWM based DAC is thought to use a 'global' feedback structure, either with signal feedback or error feedback. These will be discussed below.

5.4.2 : Case Study 1 : PWM In Noise Shaping : An Example Of Error Feedback.

As already discussed, the structure of figure 5.4.1.c offers potential to apply error feedback to a PWM based DAC and hence reduce the distortion produced by the PWM as well as the requantisation noise added by the quantiser. In chapter four, the quantiser was modelled as an additive noise source when the basic noise shaper was analysed from a frequency domain point of view. This was shown to be an acceptable model of a quantiser provided that the error signal required to null the lower bits was uncorrelated with the input. Subsequent analysis presumed this noise source to be 'white' so that the output noise power after shaping could be calculated. Nowhere in the analysis was the quantiser assumed linear. If another non-linear element is added in series with the quantiser, in particular a limited wordlength PWM, the same analysis should still be valid and the error from this should be shaped in the usual way, suppressing harmonic distortion as well as requantisation noise. As in the operation of the noise shaper, spare bandwidth will be required to move the shaped error into, so an interpolated or oversampled input is required.

The output of the PWM will be at a high sample rate so sample rate conversion must be used before calculating the error between input and output. This sample rate conversion will involve decimating filters which will have single bit input at the PWM internal clock rate and multi-bit (or floating point) output at the signal sample rate. For a PWM that can modulate b' bits, the sample rate change will be by a ratio of $2^{b'}$. ie. :

$$M = 2^{b'} \qquad \text{Equation 5.4.2.a}$$

From chapter four, it is known that the delay in processing the error should be minimised to reduce the output requantisation noise amplitude, so this decimation should be performed quickly. Multistage, multi-rate filtering is best for this, where the change by a factor of 'M', is achieved in several steps (M1,M2,...) where the product of all the sample rate changes is 'M'.

The delay of the decimating filters should equal an integer number of its output sample periods so that the NTF is not altered and simple delay elements can be used between the quantiser input and the subtraction node. With integer delay, the NTF can then be designed as a deferred filter (see section 4.5.2) allowing a zero valued coefficient for each sample period delay used. Reducing the decimation delay can be done in several ways:

1) FIR filtering, and hence linear phase, in the feedback network is an unnecessary constraint so for optimal speed of decimation, IIR filtering (and hence lower order) should be used.
2) Ideal decimating filters with 'sharp' transition bands should be avoided (large delay) so aliasing of transition bands should be allowed.
3) Multi-stage decimation, should be used, so that as much processing as possible is done at the highest rate.
4) Filters with simple coefficients should be used, taking advantage the single bit nature of the input.
5) Decimated data look-up tables to effect the first stage of filtering could be used.

Having applied the above objectives, the design of the decimating filters can be attempted using optimisation techniques [FLE63]. The case study using an 8 bit TEPWM operating at 88% modulation depth (256 of 288 output levels used), operating at 8x oversampling on 16 bit data will be used as an example. Thus, similarities can be seen to the case study of decimation for analysis in chapter three.

The output clock speed is set by the number of output levels, in this case 288, so the required decimation ratio, 'M', can be made up of the prime factors $\{2^5,3^2\}$. Linear phase in at least the passband of each decimator is required to enable reasonable precision in the calculation of the error (<1 part in 2^{16} in this case). Short FIR filtering is most appropriate for these early decimating filters so comb filters and half-band structures can be used [GOO77]. Numerous designs using different decimating factor combinations were considered and measured for their total delay. The structure with minimum delay turned out to be M_1=16, M_2=18 which is mostly because the largest delay was contributed by the final stage (which was IIR in all cases). A high processing rate in this stage enables a higher throughput of data despite the filter's magnitude response transition band being steeper (in dB/radian); this contrasts with the FIR filters where the delay is smallest for a decimation ratio of two since the order is dominated by the transition band steepness.

By knowing that only 256 of the final 288 output levels could be used, a look-up table based on a 31 tap FIR filter could be used to predict the filtered output of the PWM and first stage of decimation. This assumed that the first 16 output samples in each (input) sample period were always set, and the last 16 always clear, so that the 256 input options could be mapped directly to 256 filtered options without interference between pulses. This also allows the look-up table to run 'in parallel' with the PWM, avoiding the inherent delay introduced by the modulator†. Although processing time is

† If the feedback filter can produce data faster than the sample rate, and the next datum is available at the input, the 'next Sum' can be calculated in advance to allow more processing time still.

saved by this revised structure, the filter group delay remains the same. Using a 31 tap FIR introduces a group delay of 16 cycles (at the high rate, 101.6 MHz) and because its output is then decimated by 16 this is equivalent to a delay of one output period (at 6.35 MHz). The revised structure is shown below :

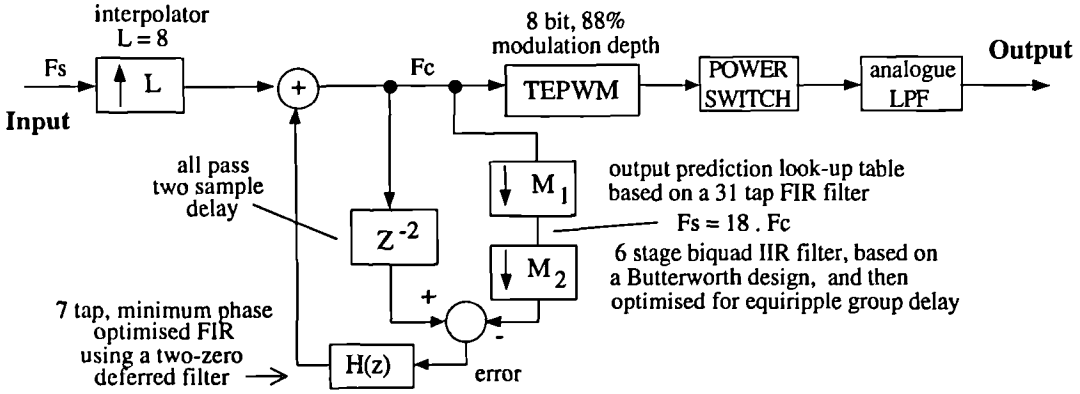


Figure 5.4.2.a : Revised Delay Compensated Closed Loop PWM DAC

The output of the first decimator is then fed to an IIR filtering stage for decimation by 18. The passband group delay of this is required to be 17 or 35, so that after sample rate conversion the overall delay of both filters is either one or two samples. This requirement arises from the optimisation of deferred filters for the 16->8 bit, 8x oversampling case. A maximum of two zero coefficients could be achieved for this case and since results from the optimisation converged repeatedly on the same solution, it is suggested that no better filters could be found.

Initial attempts to design the second stage (IIR) decimating filter by bilinear transformed analogue designs proved unsuccessful in all the cases through the group delay being too large. Chebychev I filters gave the best delay response, but only with poor passband flatness; Cauer and Bessel filters exhibited large delay and wildly varying delay, respectively; Chebychev II could not be designed to meet the anti-aliasing and delay responses simultaneously, and the Butterworth approximation could only be designed to prevent aliasing in 1/8 th. of the required band for the required delay response.

Mixed optimisation of group delay ($\approx 35\tau_s$) and a $\pi/144$ radian passband and stopband accurate to $1/2^{16}$ was attempted using the procedure of Deczky [DEC72], but the final filters could not meet all these constraints simultaneously. Assessment of the allowed delay ripple showed that the design constraints are virtually linear phase : for delay ripple that introduces the same amount of error in the feedback as passband amplitude ripple would, a full amplitude sine wave can be assessed as shown below. Although this could be partly compensated for in the all pass network, little can be done in a filter of such a low order (max. 3 taps).

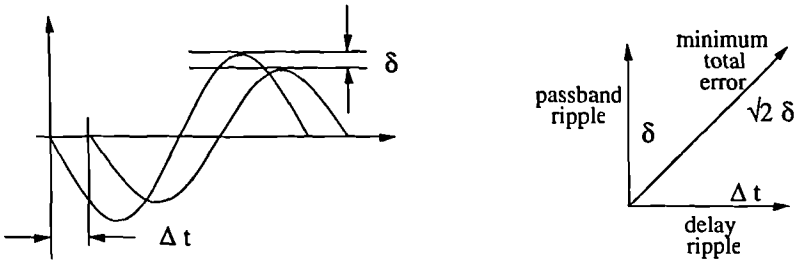


Figure 5.4.2.b : Equivalent Delay And Amplitude Error For Minimum Error Power.

Equating the error due to passband delay ripple and the error due to passband amplitude ripple,

$$\int_0^{\pi/2} \{ Sin^2(\omega t) - \delta^2 Sin^2(\omega t) \}\, dt = \int_0^{\pi/2} \{ Sin^2(\omega t) - Sin^2(\omega t + \Delta t) \}\, dt$$

Equation 5.4.2.b

yields, $\delta^2 = 4\, Sin^2(\frac{\Delta t}{2})$ which for small 'δ', implies that $\delta \approx \Delta t$

This shows that the passband requirements are not simply a near-flat passband amplitude, but also a near linear-phase passband behaviour. In light of not being able to design for this structure it was rejected and a modified structure was investigated. This is the subject of the next section.

5.4.3 : Case Study 2 : PWM in Multi-bit ΣΔ Modulation An Example of Signal Feedback.

ΣΔ modulators operate in a similar way to noise shapers, using feedback of the output instead of feedback of the quantiser error. Since the feedback error no longer has to be evaluated these structures are simpler to design than the noise shaper topology discussed in the last section. The basic block diagram is shown below marked with the variable names for the analysis which follows.

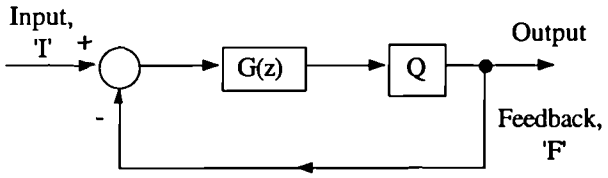


Figure 5.4.3.a : Basic ΣΔ Modulator Structure

Analysis similar to that used for a noise shaper can be used to determine the requantisation noise transfer function and the signal transfer function [HAU90]. This assumes the quantiser can be substituted by adding a noise-like signal, 'N', which happens to null out the lower bits leaving the output quoted in a shorter wordlength than the input (since its LSBs can be assumed to be zero). Although this only holds true for output wordlength greater than four bits, this is usually the case in applications using PWM. Thus,

$$STF = \frac{G(z)}{1 + G(z)}$$

Equation 5.4.3.a

and

$$NTF = \frac{1}{1 + G(z)}$$

Equation 5.4.3.b

Since the STF is no longer unity, it is common with this structure to place further filtering in front of the ΣΔ modulator, to pre-filter the signal by 1/STF, thus giving an overall transfer function of unity. G(z) is usually an integrator (ie. low pass) but can be any arbitrary shape. Providing the denominator of the transfer functions causes no poles of the transfer function to lie outside the unit circle (on the Z-plane) G(z) will remain stable.

Since the noise transfer function with this structure is different from that of a noise shaper, G(z) has to be calculated from H(z) for the NTFs to be the same as those designed in chapter four. This can only be done if '1-H(z)' (the NTF in the noise shaping case) is minimum phase since for the same NTF :

$$H(z) = \frac{G(z)}{1 + G(z)}$$

Equation 5.4.3.c

or

$$G(z) = \frac{H(z)}{1 - H(z)}$$

Equation 5.4.3.d

In order for filter G to remain stable its poles must lie inside the unit circle which implies that the zeros of the noise shaper NTF must lie inside the unit circle and thus the noise shaper NTF is only useful in the ΣΔ network if it is designed as a minimum phase NTF.

When the quantiser is replaced by a pulse width modulator, decimation is required in the feedback path. The filters required before decimation are of a low pass type which is entirely consistent with G(z), so the network can be modified, bringing G(z) back into the feedback path and taking a copy of G(z) into the signal path before the loop. Since the decimating filter and loop filter are both low pass these can be amalgamated into one filter block, and since the input is known to be low pass filtered by the interpolation before the input, the signal path copy of G(z) can be replaced by an amplifier of gain K, which was the passband gain of G(z). Finally the same gain factor can be taken out of the feedback filter, leaving a feedback filter, G'(z), with passband gain of unity, and moving the amplifier into the forward part of the loop. The revised structure is now :

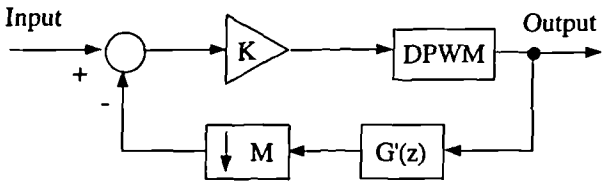


Figure 5.4.3.b : ΣΔ Based Closed Loop PWM DAC Kernel.

Some of the restrictions placed on K.G'(z) can be seen 'in advance', just as they were valid for G(z). Firstly, it is a low pass filter. Secondly, gain 'K' is approximately the amount of requantisation noise (or harmonic distortion) suppression. Thirdly, excessive delay in the loop will make it unstable. For analysis this loop can be generalised as below :

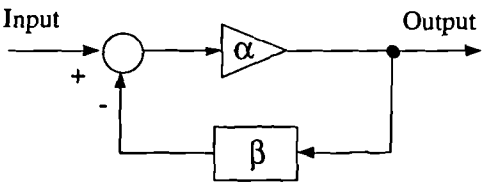


Figure 5.4.3.c : Generalised Negative Feedback Network.

By simple analysis, the closed loop gain of the above network, can be expressed as †,

$$GAIN = \frac{OUTPUT}{INPUT} = \frac{\alpha}{1+\alpha\beta}$$ Equation 5.4.3.e

for a non-ideal forward amplification, α, the influence on the overall gain can be calculated as,

$$\frac{\partial G}{\partial \alpha} = \frac{1}{(1+\alpha\beta)^2}$$ Equation 5.4.3.f

Thus, for feedback gain, β, near unity (ie. in the passband of G'(z)), and large forward gain, α, the non-linearity is reduced by a factor of α^2. Similarly, for $\beta < 1/\alpha$ (the stopband of G'(z)) the non-linearity can be seen to have *increased*. The effect of this non-linearity increase is difficult to predict without knowing the noise performance of PWM which is known only by inference from the tonal performance, or by observation of many simulations. The network of figure 5.4.3.b was programmed in PASCAL as 'sipwm.pas' and experimented with to assess the suitability of closed loop PWM DACs. Appropriate filters for use with a simple quantiser were designed and the output spectrum using 16 -> 8 bit wordlength reduction in an 8x oversampled system and for a -15 dBFS, 1 KHz tone is presented below :

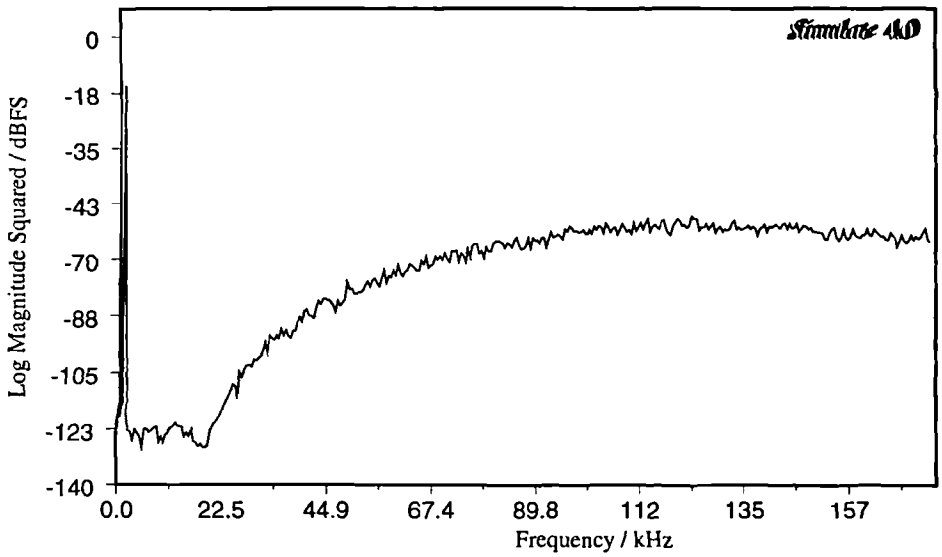


Figure 5.4.3.d : Output Spectrum of the ΣΔ Based Closed Loop PWM DAC Kernel.

From the above output spectra the network can be seen to operate as expected, with noise at high frequencies like a noise shaper. However, the structure shown in figure 5.4.3.b can only be kept stable for a simple quantiser; it becomes extremely difficult to stabilise when further delay is added by the use of filtering for sample rate change. The reason for the loop being unable to linearise with added delay in the feedback loop is simple : if the output has not yet been assessed, it cannot be compensated for. If the assessment takes too long, the subsequent inputs will have come and gone without correction of the output.

Maintaining the stability of the loop can be made easier in several ways. Firstly, the severity of the sample rate change can be reduced by using a lower PWM wordlength; this allows fewer multi-

† Appropriate scaling in the feedback decimator is required to allow for the output pulse amplitude.

rate stages to be used, and hence a lower delay. Secondly, the filtering complexity can be reduced, directly reducing the time lag but potentially allowing aliasing of the high rate output when it is down rate converted for subtraction from the input. Thirdly, the PWM output can be predicted from its input with less delay than waiting to assess the output itself (this is similar to the look-up table used in figure 5.4.2.a); unfortunately this would prevent any output stage non-linearities being taking into account. Lastly, gain 'K' can be reduced to increase the available stable input bandwidth although this damages output SNR directly forcing the use of higher PWM counter speeds.

In essence, the problem in designing the loop filtering is that a large input bandwidth cannot be accommodated with the use of large group delay filters as are required for decimation. To assess the usable bandwidth, fundamental rules of uncertainty have to be applied. This analysis is presented in section 5.4.4 and suggests that very high oversampling ratios are required, such that a better modulation type in an open loop structure would be preferable to creating an marginally stable closed loop system. This is supported by the fact that the PWMs become more linear with increased oversampling (making compensation less necessary) and that $\Sigma\Delta$ design becomes more non-linear through the low wordlength imposed by the finite limit on the PWM clock speed (and hence less stable). Furthermore, the internal computation rates in filter G'(z) become prohibitively high even with the use of systolic array processing [KNO89] or advanced predictive filtering [CRA92]. High decimation filter structures [RIL91] can alleviate the processing difficulties, but not the stability problems from the large delay required for the decimation.

5.4.4 : Finding the Minimum Oversampling Ratio for Closed Loop PWM DACs.

Using the basic network of figure 5.4.4.a, the onset of instability is known to occur when the delay in the loop is large. If sufficient phase-lag is added to the feedback signal, the subtraction at the input node is cancelled and the system will behave like one with positive feedback rather than negative.

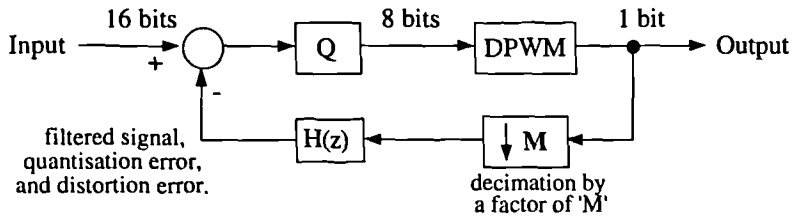


Figure 5.4.4.a : $\Sigma\Delta$ Based Loop for Assessing Closed Loop PWM DAC Stability.

The last stage of decimation is known to be that which has the steepest transition band, and thus from the Hilbert transform of the magnitude response, the largest delay. Considering this to be the dominant delay in the loop, and simplifying the frequency performance as much as possible, this will be shown to give a (generous) lower estimate of the delay required for a given decimation ratio from which the minimum input oversampling ratio, 'L', required for a given signal bandwidth, f_a can be calculated for a given dynamic range, R.

The response of this final stage of decimation must be at least that shown below :

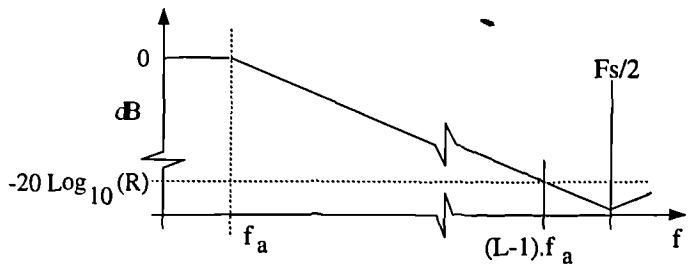


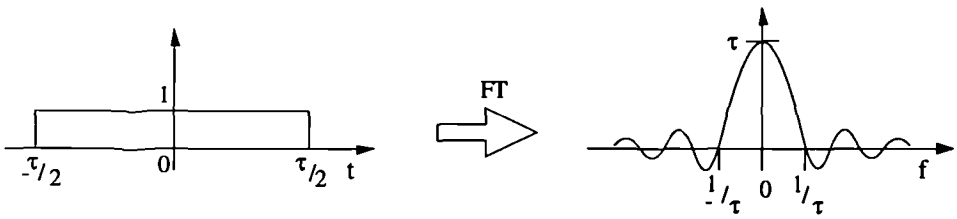
Figure 5.4.4.b : Last Stage Decimating Filter Minimum Requirements

From this the minimum transition bandwidth, Δf can be seen to be :

$$\Delta f = (L-2) \cdot f_a \qquad \text{Equation 5.4.4.a}$$

If the decimator is considered to be a 'decision box' requiring time Δt to discriminate between frequencies in the passband and stopband (ie. in the limit, frequencies separated by Δf), an estimate of the minimum theoretical delay (and hence phase lag) introduced by the decimator can be found.

By setting Δt equal to 'τ', the width of a unit height rectangular 'decision window' which would allow discrimination between two frequencies separated by Δf, to a resolution R, then the power spectral density of the rectangular window can be used to estimate Δt from 'L'. Taking the Fourier transform of the decision window, its frequency response can be found:



$s(t) = 0, \ |t| > \tau/2$
$\quad = 1, \ |t| > \tau/2$

$S(f) = \tau, \quad f = 0$
$\quad = \tau \cdot \mathrm{Sinc}(\pi f \tau), \quad f \neq 0$

Figure 5.4.4.c : The Decision Window and its Fourier Transform

For each frequency at the limit of Δf the PSD can be found by squaring the decision window, and the masking effect of each signal can be separated into an envelope and a sinusoid squared,

$$\text{PSD} = \text{ENVELOPE} \cdot \text{SINUSOID}^2$$

$$= \frac{\tau^2}{(\pi f \tau)^2} \cdot \mathrm{Sin}^2(\pi f \tau) \qquad \text{ie. E, the envelope, } = \left(\frac{1}{\pi f}\right)^2$$

Knowing the maximum value of a squared sinusoid to be one, the worst [amplitude]^2 of the spectrally spread signal at difference frequency Δf is simply the size of the envelope evaluated at Δf:

$$E(\Delta f) = \left(\frac{1}{\pi \Delta f}\right)^2 \qquad \text{Equation 5.4.4.b}$$

For a required ratio, R, between the signal and an alias,

$$R = \frac{\tau^2}{E(\Delta f)} = \tau^2 \pi^2 \Delta f^2 \qquad \text{Equation 5.4.4.c}$$

so in general, $\tau \geq \frac{\sqrt{R}}{\pi \Delta f}$, and for the last decimation stage, $\tau \geq \frac{\sqrt{R}}{\pi \cdot (L-2) \cdot f_a}$ Equation 5.4.4.d

Taking the example of 16 bit resolution digital audio data, $R \approx 10^5$, $\tau_{max.} = 100{\cdot}6 / (L-2).f_a$. Knowing that for stability, $\Delta t < 1/2.f_a$, at the minimum oversampling ratio, $\Delta t = \tau$, so

$$\frac{1}{2 . f_a} = \frac{100{\cdot}6}{(L-2) . f_a} \quad \Rightarrow \quad L = 203{\cdot}2$$

Equation 5.4.4.e

From this it is clear that large oversampling ratios are required to enable stable operation of closed loop PWM DAC structures using output decimation to produce a feedback signal.

To verify equation 5.4.4.e, a 16 bit, 20 KHz case was attempted using an oversampling ratio of 256 (the next power of two larger than the minimum required). To ensure a clock rate less then 100 MHz, 8 level PWM can be used, using 3 bit data, and decimating by 8. Since 13 bits are dropped, K=8192 and G'(z) is a minimum phase, low pass filter with a stopband width of $7\pi/8$ radians at -78 dB and a passband width of 0·001237 radians. Filters for this application were attempted through the optimisation design procedures but as the PWM wordlength is very short the filters still proved unstable in use.

Although this analysis was done using a continuous time model, very similar performance will be observed in the discrete time model since high oversampling is used in all cases. Since the envelope criterion is a worst case, stable PWM structures using feedback with output decimation may be constructed which marginally out-perform this minimum oversampling criterion. However, if high resolution systems are being considered, the value of 'R' increases. This will cause the oversampling required to force a sample rate higher than the maximum PWM counter speed, making the use of PWM impossible.

With knowledge of the PWM in use, the first stage of filtering can be done using a look-up table (ie.: prediction) reducing the oversampling required. This and other factors may alleviate the high oversampling requirement, however the digital signal processing required is extensive and required to run at extremely high speeds which may be impossible to implement.

Additional delays that have not been included, such as the early decimation stages and the adder propagation delay at the input node, and the effects of the high loop gain causing frequencies *outside* the signal band to have a gain larger then one, have not been mentioned. These serve to tighten the requirements, implying the use of yet higher oversampling ratios, so closed loop PWM is not recommended for practical PWM DACs unless emulation of the output can be used (ie. a look-up table).

5.5 : Closed Loop PWM DACs Using Output Emulation.

5.5.1 : "Noise Shaping Incorporating Correction For The PWM Non-Linearity".

By combining the best features of feedback, pre-compensation and noise shaping into one loop a new structure providing good linearity has been found [CRA92]. While design of this system is complicated because of the constraints that each part of the loop puts on each other, the performance is reported to be excellent (0.00003% THD for a 5 KHz input tone at 0 dB, noise floor < -180 dBFS) and approaches that required for 24 bit accuracy. A conceptual structure is shown below:

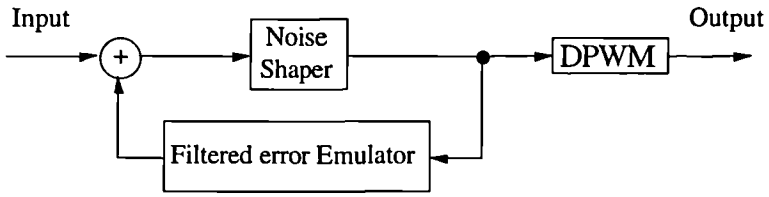


Figure 5.5.1.a : Conceptual PWM Linearising Noise Shaper

By placing a PWM 'filtered-error-emulator' in the feedback loop, the loop can be forced to produce pre-distorted noise shaped signals that for at least some of the band will produce a linear conversion after passing through the PWM. As the PWM is not in the feedback loop, decimation of its output does not have to be applied and the stability problems of the truly closed loop structure are avoided. This is very similar to the use of a PWM predictor as mentioned in section 5.4.1 but uses a non-linear feedback element based on frequency domain response matching as used in [HAW92] which predicts what the PWM would need to have done to maintain accuracy over the audio band.

The immediate drawback of this structure is that the noise shaping loop has to have a unit delay for causality, and (n+1)/2 delays to accommodate the span of the prediction filter (eg. if the predictor is based on three input samples an additional delay of two cycles is required). This can be accommodated for by limited amounts of 'procrastination' in the NTF of the shaper (similar in nature to the deferred NTF used in section 5.4). Since 'procrastination' causes large power gain increase in the NTF its application is limited to one or two zeroed coefficients and large oversampling is used. Fortunately, both a power gain reduction and an increase in the prediction accuracy can be achieved by high oversampling (64x) but computation rates for this technique are extremely severe.

It is interesting to note that correction could be based on an observation of the error signal introduced by the PWM in the time domain. The elongation of the pulse for higher input values can be viewed as the addition of energy at the end of the pulse instead of the centre of the pulse (as was originally sampled). Thus the correction required to remove this non-ideality is to add an error signal which cancels the badly placed energy and adds in that energy in the intended place. An example is shown below:

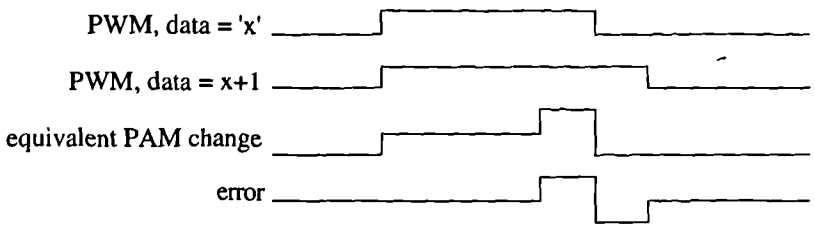


Figure 5.5.1.b : Correction Signal For Linearising TEPWM.

Since the required correction (the inverse of the error) cannot be added between input samples it has to be approximated by impulses at the nearest sampling instants. Unfortunately this makes this approach less accurate than that of the section 5.3.2. Also because this will require modification of the future samples as well as the past samples, the correction must still do some prediction.

Using frequency domain response matching the error signal for a given type of PWM can be approximated for a given accuracy over a required band. Using this technique [CRA92], a three point approximator can be derived for a trailing edged modulator with an error of 3.39×10^{-7} (if 64x oversampling is used). Discrete values from this can then be stored in ROM to convert the signal fed to the PWM into a feedback signal for the noise shaper. Although the output of the loop will be to a low wordlength, higher accuracy feedback signals are recommended with the additional use of high resolution dither to linearise the approximate nature of the discrete values stored in the compensation ROM.

Since the 'procrastinating' noise shaper has large early coefficients (after the zeroed ones), if overload of the quantiser is encountered, is is quite possible that larger errors will be generated, leading to loop instability. This is exaggerated in low bit designs (eg. 4 bit as used in the example in the literature) since the error signal is comparable in size to the output range, instantly sending the output into its saturated area. Overload prediction can be used to reduce this effect by modifying the ROM to not only predict for the PWM non-linearity, but also for the noise shaping quantiser's saturated region.

While the performance of this structure looks very promising in simulation, real time implementation requires full floating point mathematics operating at a sample rate of near 3 MHz. Since signal sources of accuracy greater than 16 bits are not currently available, this may only be viable as a future direction for PWM DACs.

5.6 : Summary.

Five fundamental structures have been examined for implementing a PWM based DAC :

(i) an open loop structure, using the best modulation type and operating conditions,
(ii) an open loop structure with various pre-distorting algorithms, many imitating natural sampling,
(iii) an open loop structure using time variant filtering to pre-compensate for PWM's distortion,
(iv) several closed loop structures using decimation of the output to produce a feedback signal, and
(v) a closed loop structure employing output emulation to produce a precalculated feedback signal.

Each provides greater linearity than PWM alone by employing some correction to cancel the PWM's inherent non-LTI behaviour.

In all the considered structures, noise shaping is employed to reduce the wordlength applied to the PWM and hence interpolation by at least four times is used in all applications. The maximum counter rate considered in the PWM itself has been limited to 100 MHz, which forces some constraints on the pre-filtering used (both linear and non-linear). Oversampling ratios beyond sixty-four times have not been seen in the literature and ratios in the range four to sixteen are considered most useful.

It should be noted that whatever noise shaping is used, the same edge accuracy as without noise shaping is always required even if the edge resolution is reduced. Furthermore, if the number of edges is increased by interpolating the input signal, the edge timing accuracy has to be increased by the square root of the interpolation factor to accommodate the increased noise contributed from having more edge errors per second [HAW92]. This effect is not severe in the audio band since interpolated signals (in contrast with oversampled signals) have the same baseband noise as the input signal, thus masking the inaccuracy caused by unimproved edge timing in higher sample rate systems.

Closed loop PWM DACs employing feedback of the error between the output and the quantised input have been shown to be unusable through the difficulties in finding the error accurately and quickly. A modified structure to avoid this problem based on a multi-bit $\Sigma\Delta$ modulator has been attempted. Even this structure has been shown to be extremely difficult to stabilise without gross oversampling ratios at which PWM is almost linear and single bit $\Sigma\Delta$ modulating converters can out-perform the PWM type in most measurements except efficiency.

Open loop compensation has been shown to be useful, at least up to the 16 bit case, for high linearity, high resolution conversion at oversampling ratios down to five times oversampling. In particular pseudo-natural PWM is suggested although time varying adaptive filtering has also been shown to be useful. Problems with sideband characteristics and intermodulation (two ways that the PWM non-linearity manifests itself) have been encountered at the 16 bit level and it is thought that this may limit the usefulness of pseudo natural PWM for higher resolution systems.

A modified feedback structure with emulation is known from the literature which offers particularly good performance [CRA92]. This is based on a particularly complex structure but it may be possible to simplify it. Use of output emulation in the closed loop structure, possibly combined with open loop pre-compensation, seems the most promising area for future research.

Chapter 6 : Noise Shaper & PWM Circuit Design.

Overview

Having looked extensively at the systems required for implementing a PWM based DAC, this chapter will examine example circuits used to verify the simulation and theory previously used. As throughout this work, the main objectives are low distortion and low noise performance, but to these, circuit simplicity (and hence reliability) will be added. Chapter 7 will deal with special aspects of power DAC design such as efficiency and special power handling considerations so these will not be looked at in depth here.

Circuits for basic single sided and double sided PWM types will be explained since more complicated modulation types can be formed from these by appropriate pre-compensation. Some examples of the simpler pre-compensation schemes for open loop architectures will also be described (eg. AOAPWM, 2SCPWM). An example of a sinusoidal noise shaper employing basic DSP functions in an application specific integrated circuit (ASIC) will be described, along with implementation details for a multi-quantiser implementation used to form a higher order loop from low order ASIC elements.

Surrounding circuitry required for low noise, low distortion performance particular to PWM based DACs will also be described. These include synchronisation circuitry between a PWM and an adapted CD player (and interpolation), the generation of a low jitter clock and a low noise power supply, and output circuitry suitable for separating the digital and analogue sections of D/A conversion.

Noise shaping will be examined first, followed by various circuits developed for PWM, clocked at up to 180 MHz. Five pulse width modulators will be looked at in total, following the course of PWM development. This starts with a basic 8 bit model, operated without oversampling or noise shaping, from which the need for a local clock was first appreciated.

The second PWM was designed to operate with oversampling, a selectable input wordlength and clock speeds of up to 144 MHz; this was used as a test-bed to choose an appropriate oversampling ratio and number of bits in the PWM. Based on RAM stored data, naturally sampled digital PWM (pre-compensated by binary search) was first demonstrated with this design.

The third PWM employs special high speed counting techniques developed during simulations of a PWM implementation as an ASIC. Fast counting and loading schemes allow this PWM to operate at up to 180 MHz, using a nominally 60 MHz CMOS logic family, and interface with an adapted CD player for real time sound reproduction. From this PWM, the need for more advanced modulation types, output synchronisation and double power supply regulation were appreciated. Using this circuit, AOAPWM and an example of time invariant pre-compensation (WAPWM) were first demonstrated.

The fourth PWM was constructed [†] using the same high speed kernel as the third, but with PAL based counting to simplify the circuitry. This was designed to be capable of 2SCPWM or AOAPWM using on-board data-bus manipulation but was only single channel.

The fifth PWM is the current 'state of the art', providing stereo double sided PWM, defined by DSP preceding it. It employs all the previously developed circuits and provides additional synchronisation schemes to form a PWM based DAC, independent of the source system.

[†] The author would like to make a special acknowledgement of the contribution of Mr. K. Davis, who carried out schematic entry, PCB layout and the programming of PAL counters for this design.

6.1 : Sinusoidal Noise Shaper ASIC Design.

6.1.1 : Design Objectives.

In order to test the use of PWM with noise shaping a basic sinusoidal noise shaper was designed †. Not knowing what combination of number of bits, oversampling ratio and order would be required forced a versatile design to meet many objectives as listed below.

1) The noise shaper should handle sixteen bit, 2's complement input data.
2) The output should be capable of wordlengths from sixteen down to single bit.
3) The error signal should also be available for testing.
4) Clipping should be included to prevent overflow and underflow.
5) The design should be capable of operating with a 256x oversampled input (11 MHz).
6) Input and output should be latched and buffered and compatible with CMOS 74HC logic.
7) Basic sinusoidal NTF coefficients should be built in, to as high an order as possible.
8) Order and quantisation level should be externally selectable.

Several technologies were considered for implementing the noise shaper including discrete circuits based on the 74HC series, Programmable Array Logic (PAL), Gate Array Logic (GAL), Field Programmable Gate Array (FPGA) and application specific integrated circuit (ASIC). Both PAL and GAL were eliminated on speed grounds since both could be bettered by discrete logic ICs from the 74HC series at that time (October 1989). FPGA was eliminated on grounds of cost since the design tools were not available. ASIC was preferred to a discrete design since the complexity and layout difficulties of the design could be easily handled by the 'floor planning' routines available within ASIC design packages. Furthermore, the SOLO design suite for European Silicon Structures' (ES2) foundry allowed schematic entry of the circuit, simulated speed testing and considerable flexibility in packaging for a 2 μm, double metallisation, N-well, CMOS process.

Using the ASIC design constraints and the design objectives, some of the initial design parameters could be established. The input and output compatibility with 74HC logic while handling up to 16 bit data at 11 MHz sample rate implied the data could not be handled in series format since the i/o speed would then be 176 MHz (74HC only operates at up to 30 MHz). Using parallel data format implied 32 i/o connections which immediately suggests that the minimum silicon area design would be limited by the number of i/o pads ('pad-limited') so the control functions were forced to be serially implemented. Using sinusoidal NTF coefficients implies that all the coefficients are simple integers and can be formed by addition, avoiding the use of a multiplier on-board the ASIC. This simplifies the design of low orders (1 & 2) but slows down the ASIC when multiple additions are required for higher order NTFs. As a result of the complexity of a higher order design, the design was limited to one providing the choice of a first or second order case.

To summarise, this implies a 16 bit parallel i/o device, operating at up to 11 MHz, selectable for first or second order sinusoidal NTF and the number of bits used in the feedback. Serial selection control and full access to the error signal were also required.

† The author would like to make a special acknowledgement of the contribution of Mr. R.G. Bowman, who carried out the schematic entry, 'floor-planning' and testing of this design.

6.1.2 : General ASIC Design Rules.

Within the ES2 library functions available for implementing basic logic functions are most of the 74 series logic family, basic gates and sequential logic functions. This enables complex logical functions to be built up from simple blocks but basic rules of fan-out capability and race hazard elimination still have to be applied on top of the Boolean requirements of the logical functions.

With the truth table of the library functions also comes timing data and 'equivalent load' information. Two aspects of the design are of particular interest after logical function. Firstly, fan-out capabilities must not be exceeded and to avoid this buffering is included for grouped signals such as clock, reset, clip overflow and clip underflow. Buffering can be inverting or non-inverting although the inverting type is preferred (if the logical requirements can be adjusted to accommodate this) since it requires fewer transistor pairs. Buffers of either type can be scaled to provide almost arbitrary load capabilities so these are left until the design is functionally completed, so that the total requirements can be measured and provided for. The second design problem is that of maximising device speed. Even if the fan-out capabilities are sufficient to ensure meeting logic threshold requirements, heavily loaded lines make their transitions more slowly (additional load capacitance) so these have to be strategically boosted in the critical timing paths (eg. sign bit extension of the latched input).

Race hazards can mostly be eliminated by conventional design approaches (adjacent groups on Karnaugh maps, latched signal paths) but care has to be taken to avoid propagation delay violations between sequential devices. One example of this is the library D-type flip-flop 'bdffn' which exhibits a D-input set up time which is smaller than the clock to output (Q or not-Q) propagation delay. This implies that if one such flip-flop's output feeds the next flip-flop's input and both are triggered by the same clock (eg. in the serial interface's shift register) the data from the first device is lost before the second device is clocked. To avoid this problem additional delay has to be added, which can be done by inserting a buffer between the two devices. Inverting buffering requires fewer transistors so the not-Q output of the first latch is used to ensure correct operation of the system.

6.1.3 : Top level Design.

From the design objectives (6.1.1) and the noise shaper structure (figure 4.1.1.a), the 'top-level' of a hierarchical design was formed as shown below.

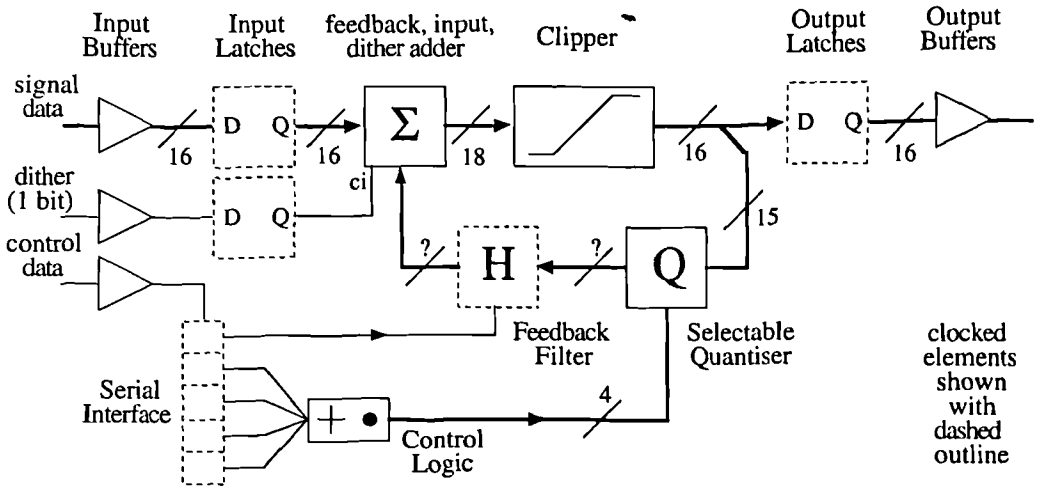


Figure 6.1.3.a : Top level Design For The Noise Shaper ASIC

It should be noted that this network varies from that shown in figure 4.1.1.a in several ways. Firstly, the calculation of the error has been removed, with the user being relied on to use only those bits that are relevant in the output. This avoids the use of subtraction in the feedback path, improving device speed (which is limited by the total feedback loop propagation delay †), and reduces silicon area (ie. cost) by removing some gates.

A second improvement in this implementation is the re-ordering of the quantiser, output and feedback. Since the error is always required for external measurements, along with the signal, the whole processed data can be fed to the output pins with only a copy being taken for that part required to be applied to the feedback filter.

The third improvement, is the use of the input adder for application of dither. While dither can be applied through node A or B (cf. chapter 4), node A requires full 16 bit addition, slowing down the loop. With the revised quantiser details, as discussed above, extra subtraction of the dither is also required from the feedback path. By using node B, single bit dither can be applied through the carry-in of the adder at the node joining the feedback and input signals. Although this restricts the dither to be single bit, it was only after the design was completed that longer wordlengths were thought to be necessary. Fortunately, single bit dither is adequate to decorrelate the first moment of the noise and hence allow proper spectral measurement without low level signal distortion or idle channel tones.

Further modifications of the system were made but these are contained in lower levels of the design and discussed with the block which they affect.

6.1.4 : Selectable Quantiser Design.

By using truncation in the quantiser and avoiding the use of dither through node A, the polarity of the error signal can be forced to be positive under all signal conditions. In a two's complement arithmetic system this will mean that the sign bit can be assumed to be clear, and this

† nb. The loop propagation delay is the sum of input to output settling times in the longest path in the feedback loop. This is not the feedback delay time as calculated from the noise transfer function (or any other feedback filters) as mentioned in chapter 5.

assists in the design of the selection of feedback bits. As discussed above, the error is continuously provided at the output, thus the function of the selectable quantiser in the feedback path can be reduced to a PCM to magnitude converter, a priority encoder, and an array of AND gates to pass the required bits of the signal. Although shown separately in the top level design, the control logic and quantiser will be dealt with together since reduction of the gate count, p-n transistor pair count, and finally the silicon area, was considered over both these functional blocks together.

By passing the feedback data through NAND gates the propagation delay of the quantiser can be kept to a minimum thus preserving the maximum system speed but causing an inversion of the logic levels (non-inverting gates require double inversion since signal amplification is always used and this is quickest if in an inverting configuration). It should be noted that since the error has been forced to be positive, after inversion this signal can be used as the negative of the error with an offset of -1. Although not used for this property in the quantiser this signal has been prepared intentionally to assist in the fast operation of multiplication by adding in the feedback filter.

In principle the priority encoder can be a cascade of logical OR gates since the state logic of the quantiser is static: its propagation delay does not affect the signal path propagation delay except in recovery from a reset and this cannot contribute to the loop propagation delay. Alternating inverted logic can be used to implement this function, but complicates the driving logic to the priority encoder, leading to an increase in the required number of p-n transistor pairs overall. When fan-out of the driving logic is also considered, the invert logic solution becomes even less favourable requiring signal boosting in several paths (ie. larger p-n transistor pairs and hence yet more silicon area).

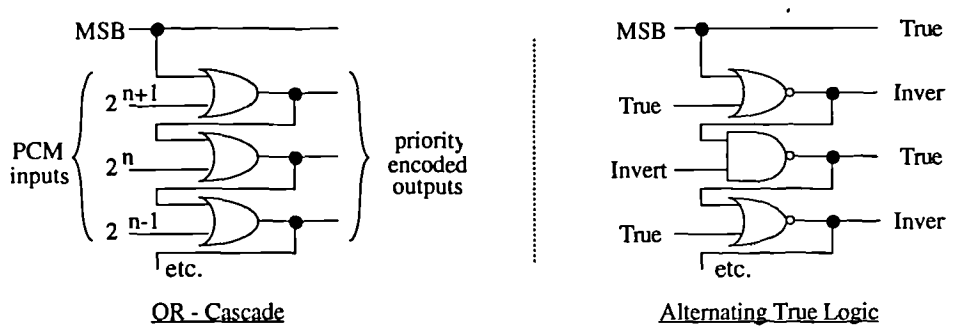


Figure 6.1.4.a ; Alternative Implementations Considered For The Priority Encoding Logic.

Choosing the OR cascade function with active high logic throughout allows simple decoding of the PCM from the serial interface which indicates the number of bits to be considered in the feedback path. Further simplification by Karnaugh map was applied and evaluated for minimum silicon area after fan-out had been satisfied in all paths from the serial interface up to the NAND functions used in the quantiser itself. The final design is implemented with only 37 gates as shown below.

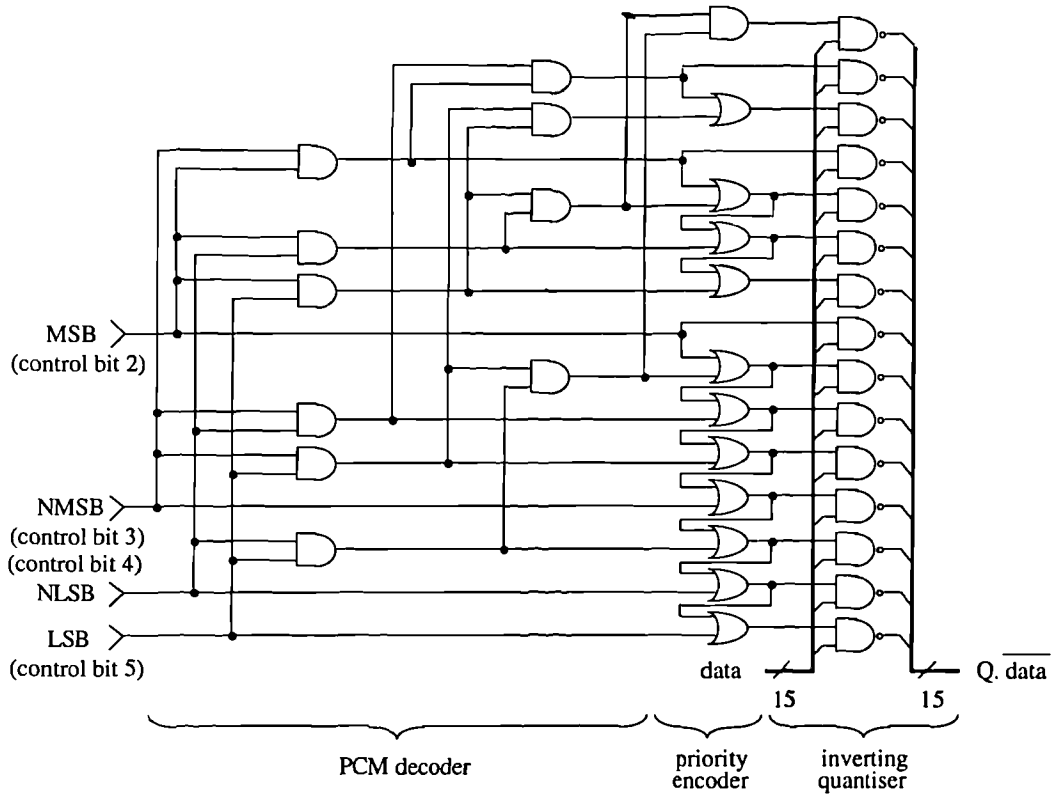


Figure 6.1.4.b : Complete Quantiser & Control Logic

6.1.5 : Feedback Multiplication Design.

By implementing only the first and second order sinusoidal noise shaper coefficients 'on-chip' the feedback can be implemented by shift and add arithmetic instead of using a multiple-add technique or even a dedicated multiplier. For first order shaping the feedback filter is simply a unit clock period delay with a gain of unity. For second order shaping, the unit cycle delayed error has to multiplied by '2' and the two cycle delayed error has to be multiplied by '-1'. Thus, depending on the order required, the output of a clocked latch has to be multiplied by two (or not) and an inverted version of the first latch output has to be latched a second time and subsequently added in (or not). This is shown schematically below:

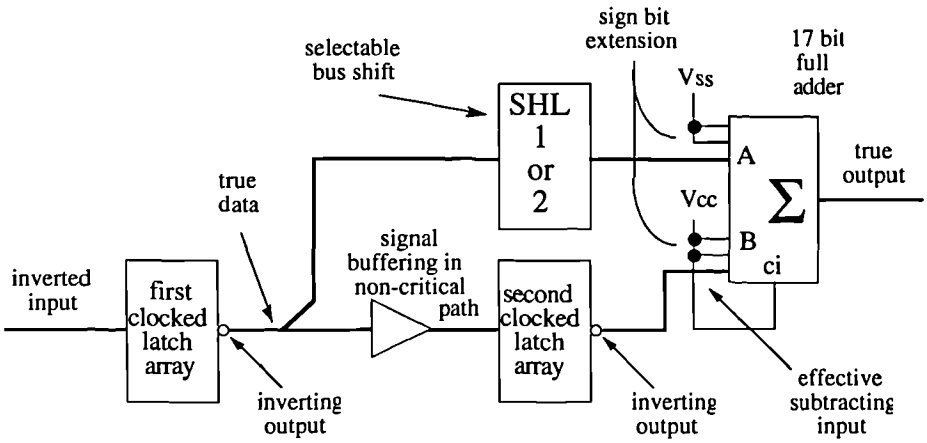


Figure 6.1.5.a : Shift & Add Implementation of the Feedback Filter.

The output of the first latch is inadequate to drive both the selectable shift (multiply) and the second clocked latch, partly because in the selectable shift the data is EXOR-ed on to two output paths (ie. loaded twice). Since the path through the selectable shift is the longer path (in terms of total propagation delay time to the next latched input), the buffering to restore acceptable fan-out from the first clocked latch array is placed only on the path to the second clocked latch array. This also ensures that the set-up times between the the two D-type flip-flop arrays is satisfied.

By using the inverting output of each array of latches the unit sample delayed data is in active high logic and the doubly delayed data is in active low logic. This enables the doubly delayed data to be subtracted by setting the carry-in of the adder (to complete the 2's complement negation of the data). When the doubly delayed signal is not required, this path can be shut off by asynchronously controlling the second latch. Since the system reset is also required to initialise the latches, the 'reset' and '2 nd. order' signals have to be functionally OR-ed although as this control input is active low, OR-ing is performed by AND-ing with inversion of the 'reset'. Since the second latch array output is inverted its 'SET' input is used in place of the 'CLEAR' input, outputting all ones when this path is not in use. Sign bit extension is required in both paths, but since the error signal is always positive the sign for both ports is known and can be wired to the appropriate state to avoid excessive MSB fan-out.

Since 15 bit data can be applied to the filter, subsequently doubled by shifting (16 bits), the maximum data value out of the filter is +65534. Similarly, the minimum data value out of the filter is -32767. In 2's complement form this range of values requires 17 bit representation, so the feedback adder's output bus was widened in preparation for this.

6.1.6 : Limiter Design.

Hard limiting (clipping) was included to avoid 2's complement overflow for large inputs. Knowing that 17 bit data can arise from the feedback filter and sixteen bit data can be applied to the input, the maximum output of the input node adder is 98302 (this includes an extra +1 for dither). The minimum signal can be calculated as -65534. These numbers imply using an 18 bit full adder and to test for overflow or underflow the sixteenth and seventeenth bits can be compared to the sign bit to make sure the output signal is still in range using the Boolean expressions :

$$\text{UNDERFLOW} = \overline{\overline{I_{18}} + I_{17} \cdot I_{16}} \quad , \quad \text{OVERFLOW} = \overline{I_{18}} \cdot (I_{17} + I_{16})$$

Equation 6.1.6.a

In the case of overflow or underflow, the output should be forced to the opposite of the sign bit which implies logically OR-ing the signal path. By using invert logic, NAND gates can be applied which can give better output drive to handle the feedback as well as the output paths and faster propagation times. Thus in the case of a detected overflow, a low level applied to the output NAND array will cause all the magnitude bits to be SET. This scheme is shown below.

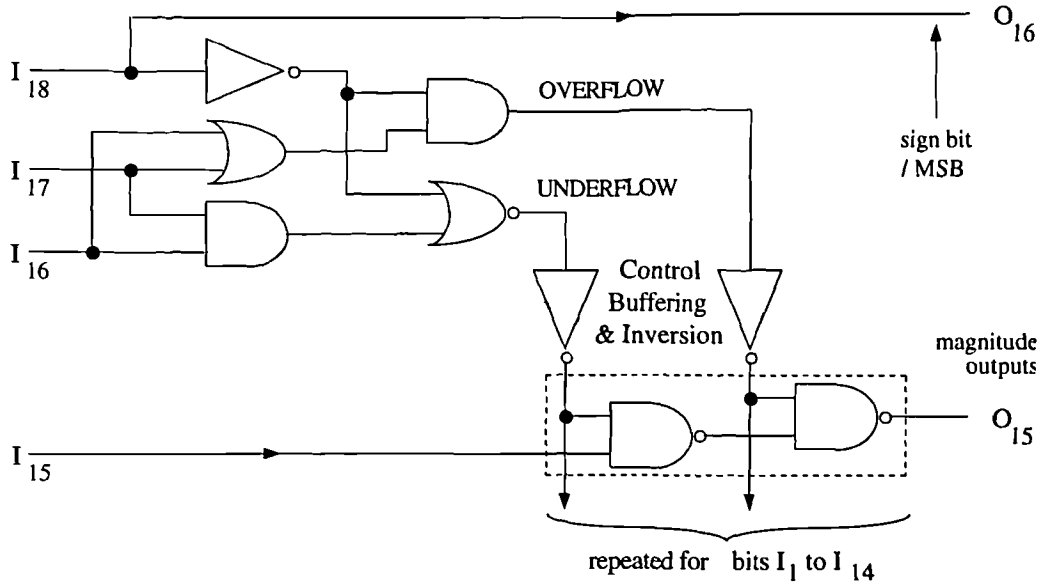


Figure 6.1.6.a : Clipping Limiter Design

Using NAND at the output implies using inverse logic in the signal path before it, hence a second array of NAND gates can be used to 'CLEAR' the magnitude bits (by SET-ing the signal before the output NAND array which invert). This also ensures that the signal logic at the input can be in active high format as supplied by the input adder†. Thus the equations of 6.1.6.a have to be inverted for use in the invert logic scheme, and buffered to drive the whole of the signal bus. Overall this adds two NAND gate delays to the loop propagation delay time and a further 37 gates to the design.

6.1.7 : Device Performance.

Having entered the design details at schematic level the device was laid out by the floor-planning routines of the design software and folded to fit into as small a silicon area as possible. 39 pins were used : 33 signal i/o ports, 2 serial data control pins (one serving as the system reset as well as the serial data window 'enable'), a clock, a clipping indicator and 2 power supply connections. As expected the layout was pad-limited, requiring only 6800 transistors. Additional grounding is recommended by the foundry between every 8-10 output pads so the last available pin in a 40 pin package was used as a central ground in the middle of the output data bus. Consecutive pin representation was used to enable simple interconnection and active high outputs were used to interface with common DSP ICs. Ceramic packaging was used which helps to reduce the capacitive loading of the connections which can become important at high speed.

After fabrication, 80% yield was achieved from 10 sample devices suggesting that the design was reasonably tolerant. With full access to the error bits, testing was relatively simple, and spectral performance of one of these ASICs is shown below being driven and measured in the digital domain by an Audio Precision System One running at 352800 kHz sample rate. An initialisation PAL was

† Since the 74 series are available in the ES2 libraries, multiple 4 bit blocks based on the 74HC283 were used for the adders throughout the ASIC. These devices include fast look-ahead carry enabling small propagation delay however they provide true logic outputs and thus the bus should comply unless full adder devices are to be designed 'from scratch'.

programmed for test purposes to load arbitrary control words. A more sophisticated version of this was subsequently produced as described in section 6.2.5 .

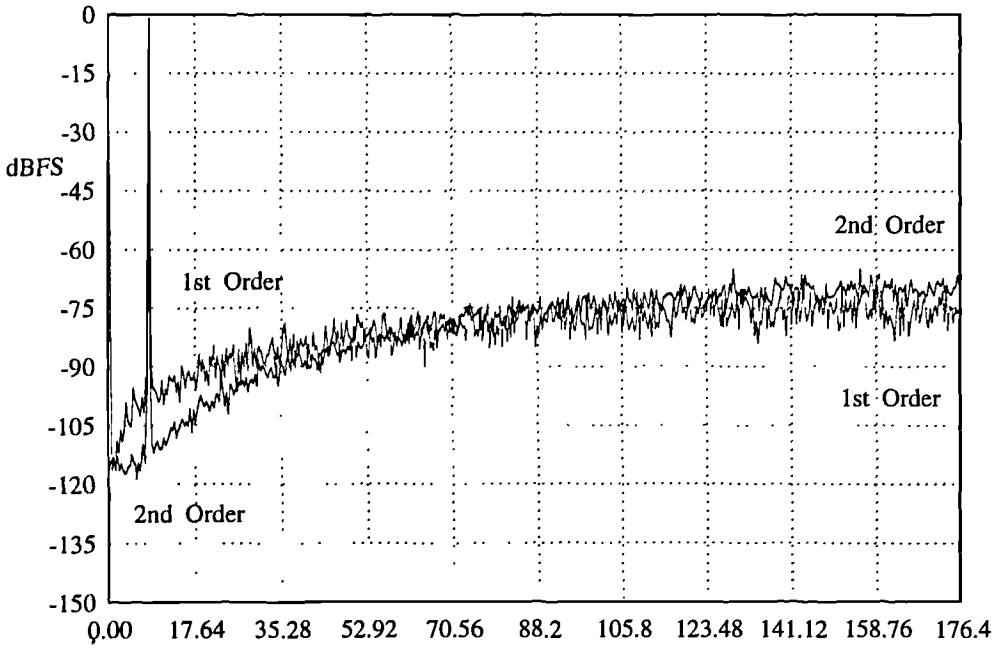


Figure 6.1.7.a : ASIC Spectral Performance Showing 1 st. & 2 nd. Order Sinusoidal Noise Shaping.

A photograph and schematics of all the hierarchical levels of the ASIC can be found in Appendix A5.

6.2 : Multi-stage Noise Shaper Design (and associated circuits).

6.2.1 : Design Objectives.

Having completed a second order sinusoidal noise shaper in ASIC as described in the last section, an open loop PWM DAC with one ASIC was used for preliminary tests at 6,7 and 8 bits and 4 with 8x oversampling. Higher numbers of bits and larger oversampling ratios could not be attempted due to the maximum clock rate in the modulator becoming unrealisable. In these tests, the noise floor near DC was low, but steadily rose at higher frequencies to give an audio band SNR in the 70-80 dB range. To produce near CD quality, higher order sinusoidal shaping would be required and in the absence of flexible DSP to do this, a cascade of ASICs was chosen to implement a high order loop.

The oversampling ratios in commercially available interpolators are mostly powers of two although PWM could theoretically operate down to 5x or 6x (depending on the modulation type used) without sidebands above the noise floor normally found in a 16 bit, 44.1 kHz sampled signal. From the distortion performance of PWM for a single tone input, oversampling at 4x was known to produce signal sidebands in the audio band so an oversampling ratio of 8x was chosen to allow standard oversampling filters to be used.

Simulation of the output performance from an 8x oversampled, 8 bit modulator showed that the low order loops (n=1,2,3) did not adequately clear requantisation noise out of the audio band. Also, high order loops were found to have such a high power gain that high frequency noise became modulated into the 0-20 kHz range (n=5,6,7). Noise sidebands were not thought to be a problem at these orders, but knowing from simulation that the best performance (at 8x oversampling) was provided by a compromise between cleared baseband and low sideband, a fourth order loop was chosen for the task.

For clock rates that can be achieved in discrete logic and using 8x oversampling, 8 bits is the maximum number that can be modulated, using a modulator operating at 90.3168 MHz. So, the final design requirements for the high order shaper were that it should operate at 352800 kHz sample rate and process 16 bit data, dropping 8 bits to provide 8 bit output for the PWM unit. It should be configurable for first to fourth order sinusoidal noise shaping and constructed in CMOS technology to compliment the existing PWM and ASIC parts with parallel i/o, similarly clocked data.

6.2.2 : Basic Structure.

A circuit was built based on the block diagram of figure 4.6.1.c based on two, 2 nd. order noise shaping ASICs surrounded by appropriate 74HC series devices to operate the filtering and addition required.

Using second order configured elements implies double differencing of the second shaped signal has to be used. Since the wordlength into the second noise shaper is restricted (8 error bits) and the gain of the second shaper is known (max. 4) the output wordlength can be calculated in advance. The additional processing delay in the lower half of the structure amounts to 2 clock cycles compared to the top half of the structure, so two latched delays are required for delay equalisation. Overflow can

occur with large signals without operation of either clipping circuit within the two noise shapers. Since the circuit was to be operated with less than full scale signals (from a CD), no additional clipping circuit was added although overflow and underflow can be detected through the carry-out of the final adder. A block diagram of all this is shown below (cf. figure 4.6.1.c).

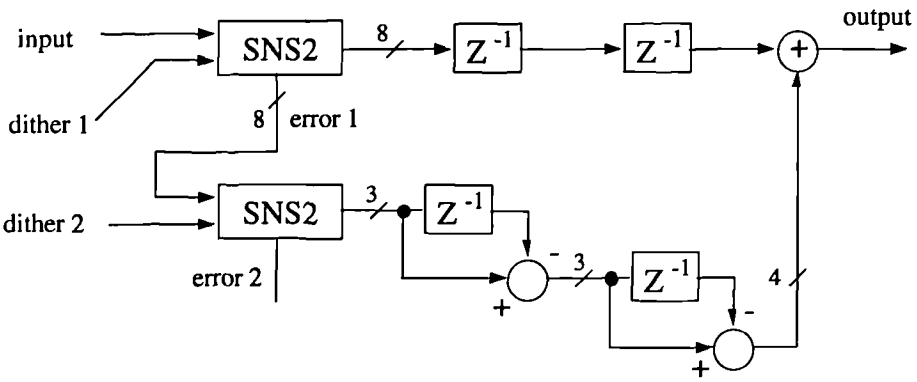


Figure 6.2.2.a : Cascaded 4 th. Order Noise Shaper

The entire circuit was built on 'vero board', without ground planes so special care was taken to provide adequate decoupling, especially for the parallel i/o devices. To reduce power supply noise, the latched delays in the top path (as shown above) were chosen to be inverting, thus reducing the current variation from these devices since consecutive outputs switch in a complementary fashion.

Both noise shapers were aligned with the LSBs representing the same signal weighting, but the top 8 bits of the second shaper were wired to ground to effect positive sign bit extension since the error from the first shaper was known to be positive in all cases.

To maintain high speed operation, fast look-ahead carry circuits were used in all the adders (74AC283) and the input and output were latched. By keeping the input and output busses physically separate from the high speed parts of the circuit, the power supply borne noise was kept to a minimum. A schematic diagram and photograph are included in appendix A7.

6.2.3 : Modified Differencer Design.

Since the differencers in figure 6.2.2.a required negation of the data in the shaped error path, the inverting outputs of the latches were used for the delay elements in the differencers. By setting the carry-in bit of the adder used in each differencer the 'invert and add one' technique of negation in a 2's complement system was carried out as used previously in the ASIC design itself.

Knowing the output of the second noise shaper could only take on values {-1,0,+1,+2}, only 3 bits are required from the ASIC to feed into the first differencer. From this the output of the first differencer can be calculated to be in the range -3 to +3, and hence only a 3 bit bus is required despite adding two 3 bit busses together. The output of the last differencer takes values in the range -6 to +6, so the bus width from the error filtering is 4 bit. Since three bit busses were used throughout the filtering, but negative values had to be handled at every port, sign bit extension was applied up to the port size of each adder. 4 bit adder blocks were used so in most cases the additional loading of the MSBs

was not severe. Obviously the output adder has to have 8 bit ports to accommodate the noise shaped signal from the first ASIC, so the filtered error has to be sign bit extended from 4 to 8 bits.

6.2.4 : Additive Dither Generator.

To create single bit dither with a rectangular PDF a maximal length shift register sequence was used. While pseudo-random noise generators of this kind do not give perfectly random performance, this was found to be adequate to remove idle channel tones from low amplitude output if used at the correct level. By using a 31 tap register, the sequence length is approximately 2.1 billion steps long before repeating occurs provided the device does not enter a recurring fixed state (EXOR feedback gives an all zero recurrent state, NEXOR feedback gives an all one recurrent state).

A standard 32 stage shift register was used, logically NEXOR-ing the 28 th. tap and the 31 st. to produce the next input value. Shift registers with eight bit capacity were used (four 74HC164 devices) but since the reset on these devices clears all the shift register contents, inversion was used in the feedback path so that the missing state is that with all the stages SET. On power up the circuit can then be held in reset (a resistor and capacitor to delay the reset pin recovery) so that once counting begins the recurrent state is avoided. Schematics for this circuit ('PRNG') are provided in appendix A7.

6.2.5 : Subtractive Dither Generator.

Using the parallel dual quantiser structure enables dithering of the signal at two points in the system, and raises the question of whether to subtractively dither or not (see section 4.6.2). An initial design for subtractive dithering was produced following the equation 4.6.2.a, and this is shown below:

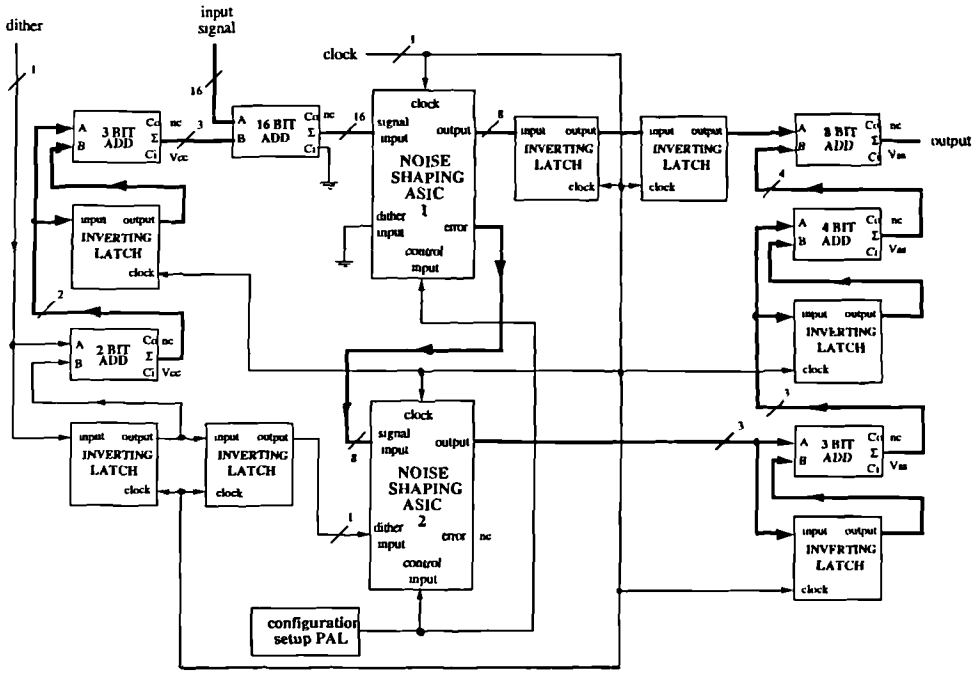


Figure 6.2.5.a : Subtractively Dithered Cascaded Noise Shaper

Since the need for subtractive dithering was not great and linearity problems in the uniformly sampled modulators were considered more significant this design was not built in hardware, but simulated (see section 4.6). Long wordlength in the dither preparation and input addition was considered a large drawback of this technique and problems have been found with the effectiveness of this technique with special signals. Its use remains an area of future research.

6.2.6 : Configuration set-up PAL.

With two ASICs on the cascaded 4 th. order noise shaper board and an alternative 2 nd. order ASIC embedded on the 3 rd. design of DPWM (see section 6.4.3) a need arose for a loading circuit to setup the serial ports of the noise shapers. Since the configuration is simply a preset combination of the order required and the number of error bits to consider, a switch array was used to define states to a mixed sequential / combinatorial PAL, programmed to output the states in serial form at the clock rate after an initialisation pulse from the user. A second feature was built-in, to ensure only complete control words were admitted to the ASIC serial port : a reset signal was produced from the PAL which ensures that the transmission only stops after a control word has been finished. This effectively debounces the user initiation and prevents corrupted data in the serial word. A state diagram for the load sequence is shown below:

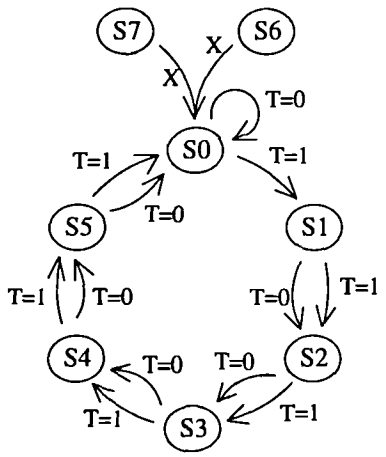


Figure 6.2.6.a : State Diagram Of The Parallel To Serial Converter (ASIC Loading PAL).

From the state map, the table of next states was calculated :

State Number	Binary Allocation			Next States		Next states, in binary : T=0,			T=1,		
	Q3	Q2	Q1	T=0	T=1	Q3	Q2	Q1	Q3	Q2	Q1
S0	0	0	0	S0	S1	0	0	0	0	0	1
S1	0	0	1	S2	S2	0	1	1	0	1	1
S2	0	1	1	S3	S3	0	1	0	0	1	0
S3	0	1	0	S4	S4	1	1	0	1	1	0
S4	1	1	0	S5	S5	1	0	0	1	0	0
S5	1	0	0	S0	S0	0	0	0	0	0	0
S6	1	0	1	S0	S0	0	0	0	0	0	0
S7	1	1	1	S0	S0	0	0	0	0	0	0

Figure 6.2.6.b : Next States Table For The ASIC Loading PAL.

By ensuring the next state was only conditional for the rest state (dependent on the initiation signal T), the sequence could be forced to not stop until a complete word had been transmitted.

From the next states, the sequential and combinatorial logic equations were calculated for the two required signals, serial data ('SD') and load enable ('LE'). Spare inputs were available, so an inverter was added to support crystal operation if asynchronous loading was required ('XOUT', 'XIN') and an inverted version of the serial data was provided to balance line currents if required ('NSD'). The unused flip-flop ('Q4') of the four available in the chosen target device (16R4) was used to indicate if the PAL had been in an illegal state, to assist debugging at startup. Gray coding was used in the binary state numbering to ensure that race hazards could be covered by alternative logic in the combinatorial equations. Active high and active low initiation was built-in ('T1', 'T2'), using the logical OR of the two active states to trigger a serial transmission so that power up reset and user override could be used if required. The sequential and combinatorial equations are listed below:

$$Q1 := \overline{Q3} \cdot \overline{Q2} \cdot (Q1 + T1 + \overline{T2}) \quad [6.2.6.a] \qquad Q3 := Q2 \cdot \overline{Q1} \quad [6.2.6.b]$$

$$Q2 := \overline{Q3} \cdot (Q2 + Q1) \quad [6.2.6.c] \qquad Q4 := Q3 \cdot Q1 \quad [6.2.6.d]$$

Equations 6.2.6.a-d (sequential logic)

$$\overline{LE} = (Q3 + Q2 + Q1) \cdot (\overline{Q1 \cdot Q3}) \quad [6.2.6.e] \qquad \overline{XOUT} = \overline{XIN} \quad [6.2.6.f]$$

$$\overline{SD} = CB1 \cdot (\overline{Q3} \cdot Q2 \cdot Q1) + CB2 \cdot (\overline{Q3} \cdot Q2 \cdot Q1) + CB3 \cdot (\overline{Q3} \cdot Q2 \cdot \overline{Q1}) + CB4 \cdot (Q3 \cdot Q2 \cdot \overline{Q1}) + CB5 \cdot (Q3 \cdot \overline{Q2} \cdot \overline{Q1}) \quad [6.2.6.g]$$

$$NSD = CB1 \cdot (\overline{Q3} \cdot Q2 \cdot Q1) + CB2 \cdot (\overline{Q3} \cdot Q2 \cdot Q1) + CB3 \cdot (\overline{Q3} \cdot Q2 \cdot \overline{Q1}) + CB4 \cdot (Q3 \cdot Q2 \cdot \overline{Q1}) + CB5 \cdot (Q3 \cdot \overline{Q2} \cdot \overline{Q1}) \quad [6.2.6.h]$$

Equations 6.2.6.e-h (combinatorial logic)

The operation was logically simulated (not taking timing into account) and a device was programmed for the task (767 fuse elements were required). Programming files of logic equations and simulated performance are included in appendix A5.

6.2.7 : Spectral Performance.

With the whole circuit together, containing the additive dither generator, the configuration PAL and the fourth order cascaded noise shaper, spectral tests were performed using the board configured as a 1 st., 2.nd., 3 rd. & 4 th. order loop in turn. 16 bit input data was used in each case, and 8 bit output data, both sampled at 352800 kHz. The signal was generated and measured purely in the digital domain by an Audio Precision System One which was also used to perform 16384 point FFT with a Blackman Harris four term window. These spectra are shown below.

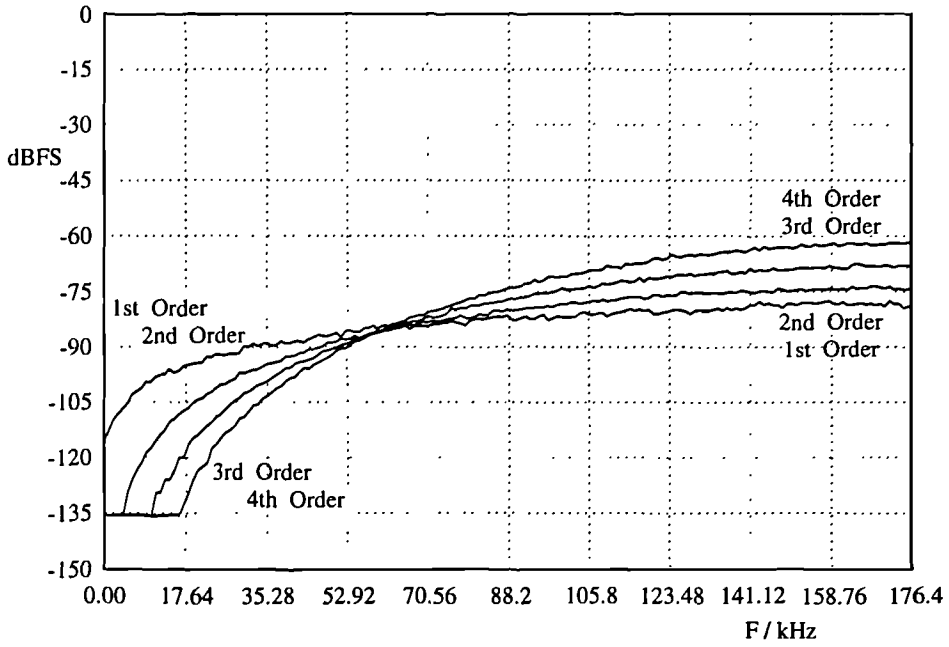


Figure 6.2.7.a ; Spectral Performance Of The First To Fourth Order Sinusoidal Noise Shapers.

From this, the improved SNR in the audio band that can be gained from using a higher order noise shaper is clear, as is the high noise gain 'penalty' at high frequency (cf. figure 4.2.4.a & b and figure 4.3.1.a).

6.3 : Clock Design And I/O Circuits.

6.3.1 : Fast Clock Generator Design.

There are three approaches for producing the fast clock required to drive the counters used in the DPWM. The first involves taking some low frequency from the signal source such as the sample rate, or if the signal is available in serial form a product of the sample rate, and frequency multiplying using a phase locked loop (PLL) to produce the desired counter rate. In the second and third approaches the high speed clock is generated by a local oscillator; the second approach then uses a large data buffer to accommodate any difference between the local data rate and the source data rate. The third approach uses time varying sample rate conversion (ASRC, [ADI93]) to synchronise with the source.

PLL frequency multiplication produces a less stable high speed clock since error in the PLL itself causes frequency modulation (FM) of the output frequency. The higher the frequency ratio in the PLL, the larger the FM of the output; to keep the output as stable as possible, the highest speed clock that is available from the source device should be used. In the first of the DPWMs built (see section 6.4.1), the data source was an 8 bit EPROM programmed with a sinusoid data set which was read repeatedly. Data was produced in parallel format at 10% above the Nyquist sample rate, so the fastest clock available was 44.1 kHz which was multiplied by 256 (to 11 MHz) to produce the high rate clock. This proved a good demonstrator of the severity of jitter introduced into the output by FM of the PLL error and steered subsequent experimental design away from this approach.

Commercial designs mostly use this technique since different signal sources operate with a different frequency for the fastest available clock. Clock recognition is always required and it would be possible to use this to setup the appropriate PWM and noise shaper parameters. Alternatively, a rational interpolation ratio could be switched in, so that the PWM DAC always operates under the conditions for which it was designed.

Various clock rates are in use commercially: CD players use 44.1 kHz as the basic sample rate for reading signals, DAT uses 48 kHz, DAB uses 36 kHz; three further differences arise:

1) different oversampling ratios (eg 4x for a multi-bit DAC up to 256x for a bitstream DAC),

2) different serial data grouping (left & right separately or together, maybe more in the future),

3) different serial data format (16,20 or 24 bit data frame and maybe additional parity bits).

The fastest available clock is usually the 'bitclock' of serial data communication between the DAC and the signal interpolator for it. This can be up to 384 times higher than the stored sampling rate (24 bit data frame for left and right channels at 8x oversampling). This enables the PLL ratio to be far less (eg. 6x) and the fourth DPWM is designed to be used with this (no on-board clock circuitry).

An alternative fast clock generator was used for the second and third DPWMs so that jitter introduced by PLLs could be eliminated from experimental measurements. In the second DPWM, flexible clock speeds were designed for up to 144 MHz, cycling through RAM-stored data repeatedly. This limits the signals to be periodic within the length of the RAM and forces the quantisation noise in the signal to be harmonically related to the signal. In the third DPWM, a CD source was interfaced to, so the clock frequency became fixed. Designing for this is the subject of the next section. In the fifth DPWM design ASRC was used so that the PWM and noise shaper were locally synchronised.

6.3.2 : Crystal Oscillator Design.

Using a modified CD player as a signal source required that the clock rate for the CD player should be derived from the fast clock so that the input and source data rates were the same. This does not force the same clock phase on both sides of the input interface but this problem can be solved with a small amount of data buffering (see section 6.3.4). A player with a built-in interpolator was used initially which required a 384x Fs clock (16.9344 MHz) with 50% duty cycle. Designing for 8x interpolated, 8 bit modulation (2048x Fs) implied that a non-integeric ratio (5.333x) existed between the fast clock and the clock required to run the CD player. To solve this a higher clock rate was chosen and any spare fast clock periods in the DPWM carrier period were allocated to guard bands. To maintain the highest modulation depth, the smallest fast clock rate which was the next highest integeric ratio with the CD driving clock had to be calculated (ie. 6x). This implies that the clock rate for the DPWM has to raised to 6 x 16.9344 MHz (101.6064 MHz) and a divide by 6 circuit which can operate at this frequency to produce a 50% duty cycle signal has to be devised (see section 6.3.3).

For maximum stability a quartz crystal based oscillator was chosen, but the thickness of crystals that mechanically resonate at c.100 MHz is so small that they are very sensitive to mechanical damage. To avoid this and the production cost of such a crystal, a specially tuned crystal was used for operation at 20321280 Hz and a circuit was built to encourage oscillation at the fifth harmonic of its resonant frequency.

Operating the crystal at its fifth harmonic requires suppression of the resonances at other odd order harmonics so additional filtering is required in the feedback of the oscillator circuit. A resonance in the oscillator gain at the required frequency can be introduced with a second order filter (L-C) but the quality factor (Q) of such a filter has to be as small as possible to avoid steering the crystal off resonance with temperature and drive current variation. Providing adequate gain to encourage the fifth harmonic oscillation, and adequate Q to suppress the third harmonic (and fundamental) also allows the seventh harmonic to resonate because the resonant peak amplitudes decay in an approximately 1/f manner. Additional notch filtering at the seventh harmonic has to be added to isolate the fifth harmonic. Even order harmonics need not be suppressed since the crystal has electrical contacts at both ends, forcing nodes at the fixing points and hence symmetry in the mechanical oscillation. An oscillator with the appropriate gain-frequency characteristic for this purpose was designed and is shown below:

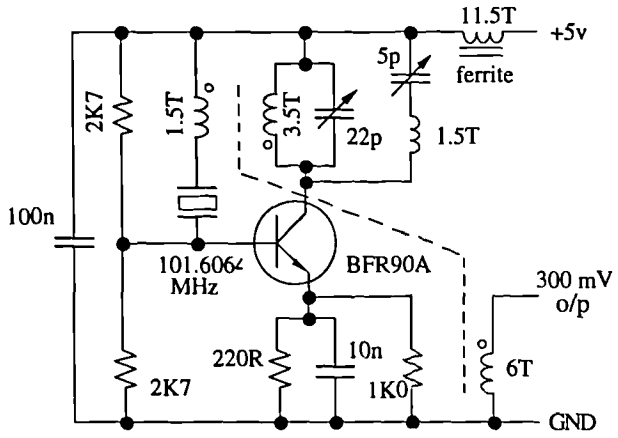


Figure 6.3.2.a : Crystal Based Oscillator For Stimulating The Fifth Harmonic Of Resonance.

This circuit is curious in its use as an oscillator, since the transformer coupling used in the feedback would suggest that negative feedback has been applied, however the phase rotation in the transistor (its 'propagation delay') is used to complete a positive feedback loop. Transformer coupling also allows easy control of the feedback gain without altering the transistor biasing and simple tapping off of a signal for subsequent amplification.

Following appropriate selection of the required harmonic, the signal has to be amplified to allow both the driving of 50Ω lines and low dissipation power in the crystal to allow high spectral purity. To maintain this spectral purity the signal should not be amplified more than necessary since clipping the signal at any later stage introduces saturation in that device with ensuing hysteresis and hence phase modulation of the clock (this leads to more jitter in the DPWM output). A two stage class A amplifier was used, taking the signal away from the oscillator in a coaxial line to avoid parasitic interactions, and using a common collector configuration followed by a common source configuration (based on a FETlington device). A 6v peak to peak output signal was derived because of some resonant action in the output devices; this was centred at 2.5v at an impedance of 33Ω, adequate to drive a 50Ω line for logic devices. A full schematic and photograph are provided in appendix A7.

6.3.3 : High Speed Clock Division Circuitry.

Having obtained a high frequency clock from the crystal oscillator (as described in the previous section), the CD clock needs to be derived from this by dividing the clock by a factor of six. The output clock signal to the CD player needs to be 50% duty cycle since rising and falling edge triggered devices of the high speed CMOS family (custom '74HC' CMOS) are used internally. These devices have propagation delays measured at near 25 ns. and hence cannot be used sequentially with clock pulse widths less than 30 ns. in either the true clock or inverted clock. The period at 16.9344 MHz is 59 ns. so there is not a great deal of flexibility in the duty cycle of the clock. To achieve the 50% output duty cycle, the division by six has to be split into division by three followed by division by two. This forces the division by three to be operated at 100 MHz so the advanced CMOS series was used. Asynchronous operation with a minimum of connections to reduce fan-out in the critical timing paths was chosen. Using the fastest AC series device, the 74AC109 which is capable of toggling at 210 MHz, forced a J-K flip-flop design using NOT-K; a state table for this was drawn up :

state	Q1	Q2	J1	K1	$\overline{K1}$	J2	K2	$\overline{K2}$
0	0	0	0	X	X	1	X	X
1	0	1	1	X	X	X	1	0
2	1	0	X	1	0	0	X	X
3	1	1	X	X	X	X	X	X

Figure 6.3.3.a : State Table For The High Speed, Divide by Three Circuit.

Of the several designs possible, those which did not have race hazards or start up problems were sought. For minimum fan-out on every feedback signal and ensuring that a suitable 'end of count' signal was provided for driving load circuits in the PWM, the following combination must be used:

$$J_1 = Q_2 \ , \ \overline{K_1} = \overline{Q_1} \ , \ J_2 = \overline{Q_1} \ , \ \overline{K_2} = \overline{Q_2}$$

Equation 6.3.3.a

If the 'illegal' state (s=3) was to occur with this configuration, both Q1's and Q2's next state is zero so the circuit will return to normal operation in one cycle at startup. The output is a 33% duty cycle pulse in the last 1/3 of the repetition period and both the active high and active low versions have only one feedback signal so that loading of subsequent circuits is best spread by using one of each polarity. The schematic for the fast divide by 3 is shown below but the full divide by six circuit would require this to be followed by a further dision by 2.

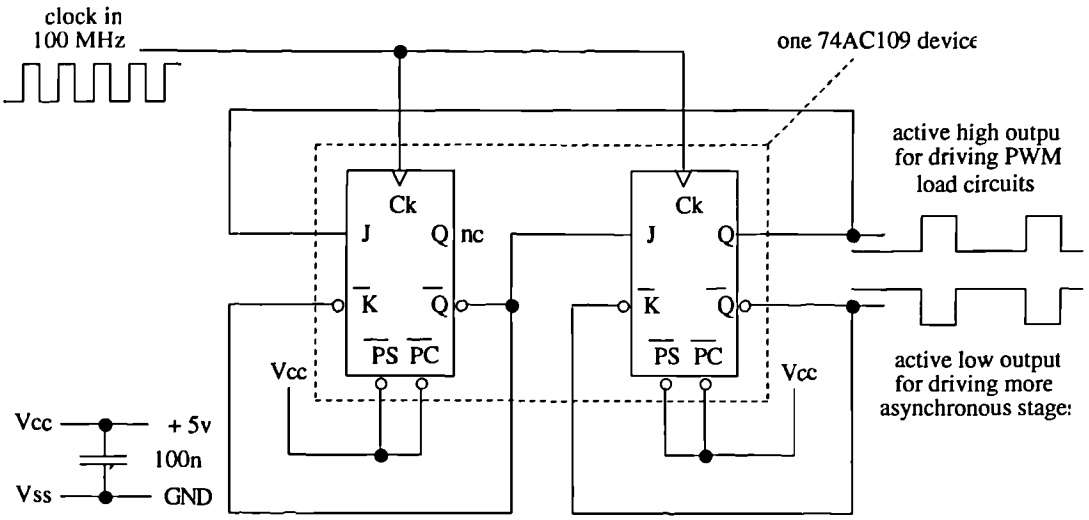


Figure 6.3.3.b : High Speed Divide By Three Circuit.

6.3.4 : Input Circuits.

To force the CD player to operate as a slave device, clocked by the externally provided clock from the fast clock division circuits, the CD player was modified. The external clock signal was fed into the internal oscillator circuits and being at a lower impedance than the internal clock (and a fractionally different frequency than the built-in crystal) this signal can dominate the crystal oscillator and hence supersede the internal circuits after powering up. If the external clock is absent for a few cycles (eg. the external clock is not connected or has been stopped in some way), the internal crystal will begin to oscillate and then being at its resonant frequency (and at a low impedance) this clock will continue and cannot be disturbed. In this way an external clock can be functionally OR-ed with the internal clock without switching circuitry and without clock confusion. The interpolated data from the internal DAC ICs was taken out through buffering so that left and right channel serial data corresponding to the analogue outputs from the player were available. This not only allowed the serial data to be taken out of the CD player without loading the internal lines (because of buffering), but also allowed direct comparison of analogue and digital signals simultaneously.

Since the CD player was not earthed, the internal connections that had been brought out had considerable 50 Hz analogue signal added to the serial data, word clock and bit clock. Optical isolation was added in the lines between the CD player and the PWM DAC input to remove this interference and consequently eliminate any chance of signal loops to pick up other analogue signals or RFI. Since the external clock signal was operating at 16.9344 MHz care had to be taken to ensure this signal remained

properly 50 % duty cycle despite the hysteresis in the isolation. Additional biasing and partial speed-up capacitors were used in the LED drive circuits until the output was seen to be close to 50% and heavy decoupling was used at the outputs to reduce interference in the line.

After some use of this interface, the internal interpolator was replaced by an external interpolator. This meant that the bit clock, word clock and serial data lines were changed internally to those before the interpolator. The serial data format changed as well as the sampling rate because the left and right channels split after the interpolator, so fewer lines were required after this change.

The internal interpolator (CXD2551) was observed to suppress the 88.2 kHz and 176.4 kHz spectral replicates of the baseband by only 40 dB (worst case). A spectral plot of its frequency response derived by stimulating the filter with an impulse is shown below:

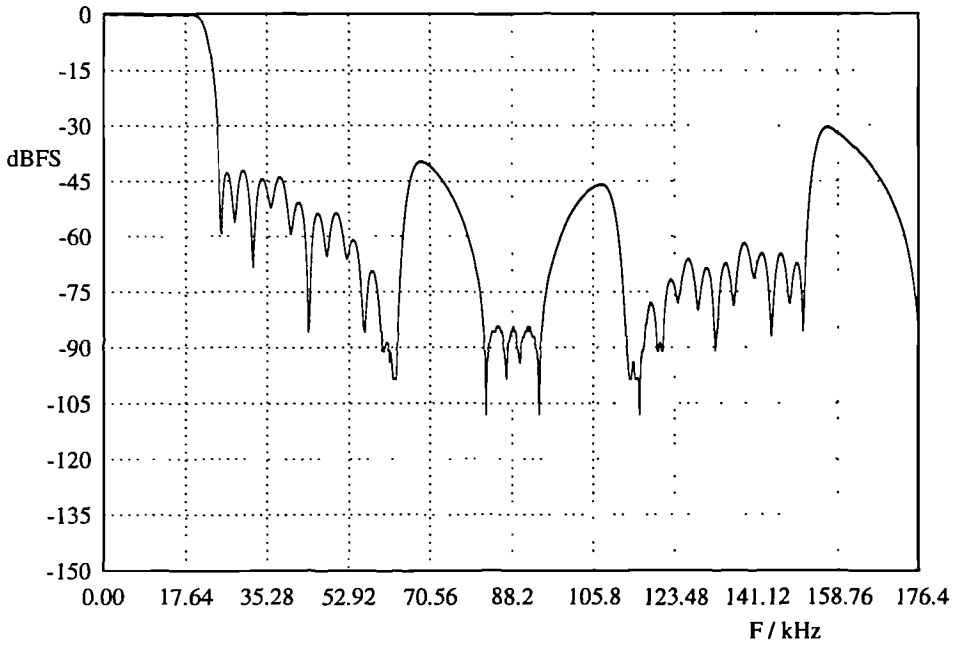


Figure 6.3.4.a : Frequency Response Of The CXD2551 8x Interpolating Filter.

In most commercial multi-bit DACs (eg. Burr Brown's 'PCM56' as used in the CD player) the spectral replicates do not interfere with the signal and provided that they are low enough to 'disappear' after slow roll-off analogue low pass filtering, the subsequent amplifier and loudspeaker will not be harmed. Typical output performance for a -6 dBFS tome at 1 kHz is shown below :

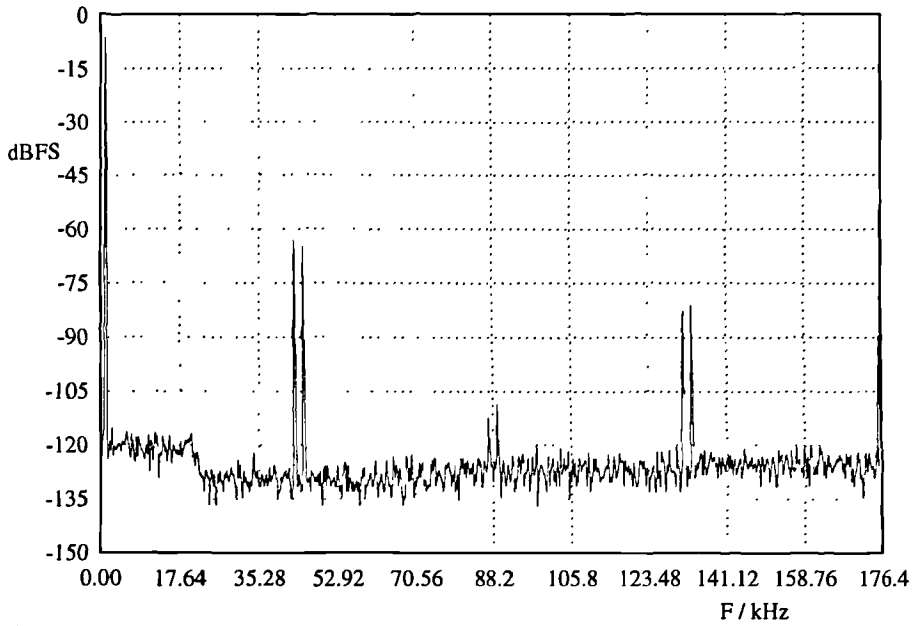


Figure 6.3.4.b : Output Spectrum From The CXD2551 8x Interpolating Filter For A 1 kHz Input.

The frequency of the spectral replicates is too high to hear so in most cases such an interpolator is adequate. In PWM DACs however, high frequencies in the input are modulated so that they may re-appear in the audio band; linear time invariant techniques (such as analogue low pass filtering) cannot remove this interference. To avoid this source of error Burr Brown's DF1700, 8x interpolating IC was added externally. This has a far better frequency response (>-110 dB suppression outside the audio band) and was used in subsequent high resolution measurements. The output spectrum for this interpolator is shown below with the same test conditions as used in figure 6.3.4.b.

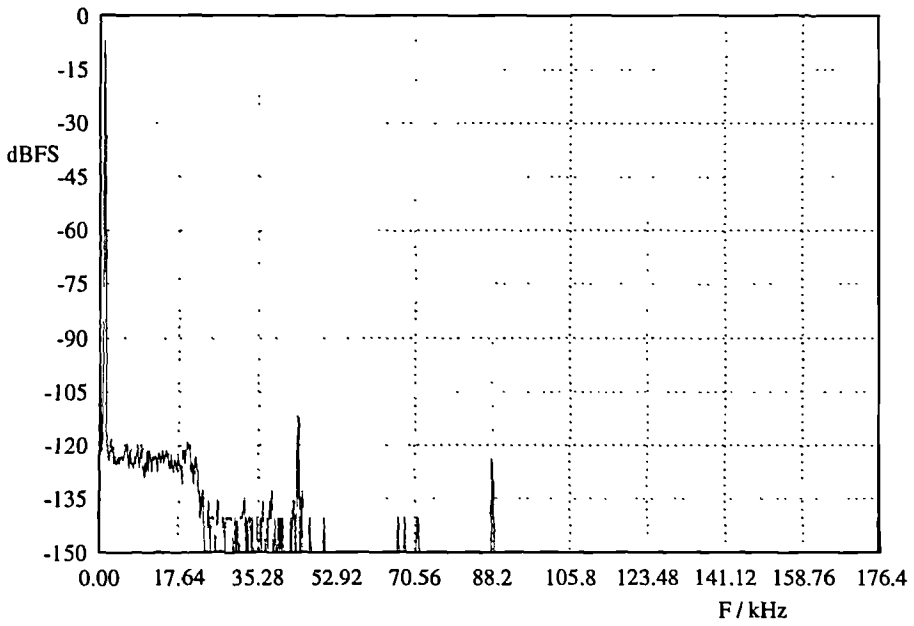


Figure 6.3.4.c : Output Spectrum From The DF1700 8x Interpolating Filter For A 1 kHz Input.
(nb.: the 45 dB improvement in the spectral replicate suppression near 44 kHz)

The output of the DF1700 is 20 bit, in serial format, with separate left and right channels. A diagram of its circuit configuration is in appendix A7.

This signal was converted to parallel format before first-in, first-out buffering (FIFO) with the CD40105BE device, to drive the noise shaper or pre-compensator circuits. FIFO buffering was added because the sampling instants at the input and output of the interpolator have an arbitrary phase at start-up. Since 8 data samples are output for each input sample, the clock phase cannot be predicted unless the 'left-right' clock of the DF1700 is used as the sample rate clock in the PWM DAC (this clock has an awkward phase with respect to the load time used in the PWM). On top of this, for some experiments DSP cards for pre-compensation were run asynchronously, leading to a varying interrupt-to-response delay time depending on the state of the software 'wait loop' at the time of interrupt.

By inserting 16 cycles of FIFO both these problems can be avoided with the interpolator providing data when it is ready and the PWM DAC reading the data when it wants it. An 8 stage shift register was used to initialise the FIFO so that output reads only start 8 cycles after input data is admitted. This circuit is shown below:

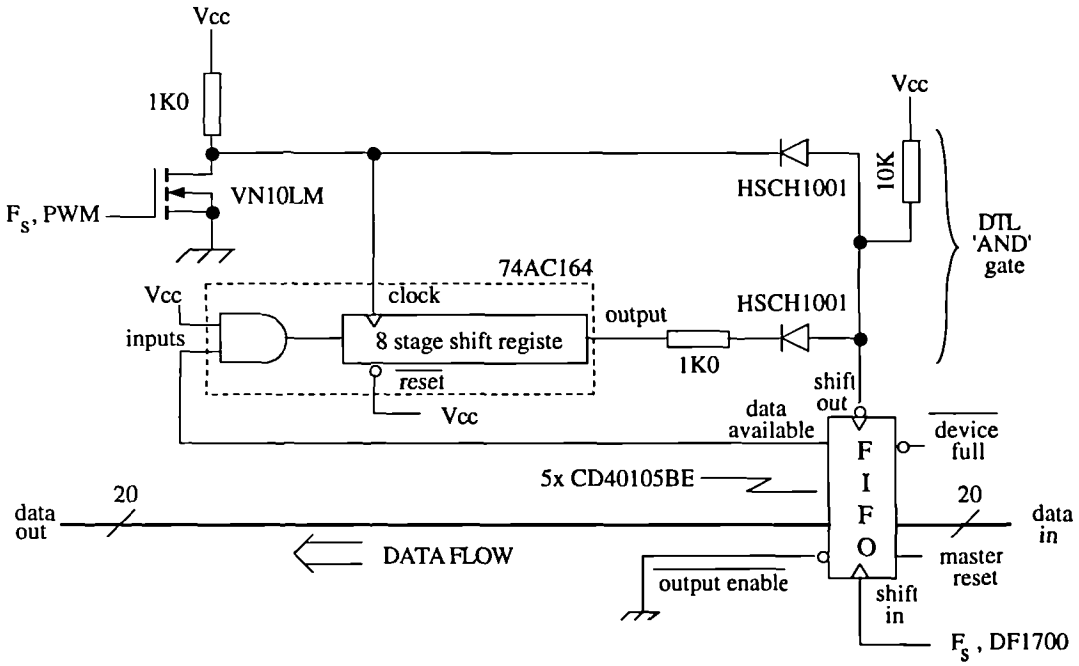


Figure 6.3.4.b : FIFO Clock Control Circuitry

In this way the FIFO operates in the middle of its range, being read from and written to at the same rate but asynchronously. The output lines are also series terminated with 150Ω resistors to reduce circulating currents in the data bus. A schematic of all these input circuits is provided in appendix A7.

6.3.5 : Output Circuits.

To reduce audio frequency power supply voltage variation on the PWM PCB, current fluctuations should be reduced to as small a level as possible so that the board operates as a constant current device. Constant current operation can be achieved by using complementary signals but this is not always practical (or necessary). Although local switching transients can be virtually eliminated by using low impedance supplies and heavy decoupling (eg. continuous power planes and 100 nF/IC), low frequency current fluctuations cannot be handled in the same way.

Internal driving impedances are near 200Ω (and dynamic) whereas external impedances are typically 50Ω (and resistively terminated) for high speed coaxial lines, so reducing the variation in the current supplied to external loads is key to reducing current variation to the PWM and thus voltage variation due to inadequate supply level control. To enable this complementary outputs were provided and a re-synchroniser was added to perform a second stage of latching (running at the fast clock rate).

Since the re-synchroniser operates on complementary signals and produces complementary output the current drawn from its supply (which was separately regulated) becomes virtually constant, depending only on load impedance variations. This eliminates phase modulation of the output as the logic threshold moves with the power supply level and ensures minimal amplitude modulation of the output pulses. A dual J-K flip flop was wired up to operate as a pair of D-type flip flops, and their clocks were provided before the connection to the PWM itself. The layout has been photographed for this and is included in appendix A7.

The output of the re-synchroniser is so 'clean' that it can be used to directly drive a signal level low pass filter for assessment of the PWM DAC without an output power switch. To avoid loading the logic to a point at which the switching exhibited hysteresis the output signal was fed through a potential divider of total impedance 317Ω. A 50Ω tap was taken from this to supply a symmetric seventh order analogue low pass filter with a passband of approximately 50 kHz (virtually linear phase up to 20 kHz) and a suppression of 125 dB at 352800 kHz. The unused line was similarly loaded to maintain the constant current operation of the re-synchroniser with 298Ω to give good matching near 1 kHz for test purposes. The measured amplitude response of this filter is shown below but full simulated performance for this is presented in chapter 7 :

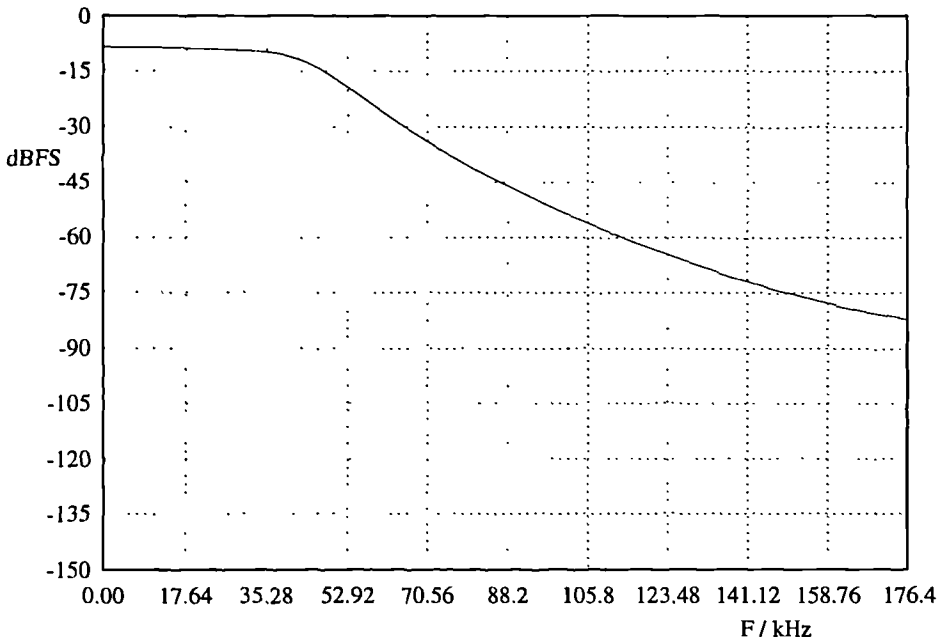


Figure 6.3.5.a : Measured Amplitude Response Of The Seventh Order Signal Level Output LPF.

As can be seen above, the output LPF has 8 dB loss in the passband, so output signals are small and need careful treatment to avoid measurement errors. Also, an large passband width has been used to avoid phase distortion of the signal, so noise from noise shaping will be present in the output.

To ensure minimum power supply borne noise, double linear regulation was used along with a star earthing layout and power supply filtering. A bifilar wound inductor was used to suppress differential mode currents and subsequent supply smoothing was used to remove audio frequencies from the power supplies as well as local logic decoupling. Single regulation was found to provide about 80 dB of supply ripple rejection although this was totally eliminated after double regulation and filtering as can be seen from the spectral plots shown below:

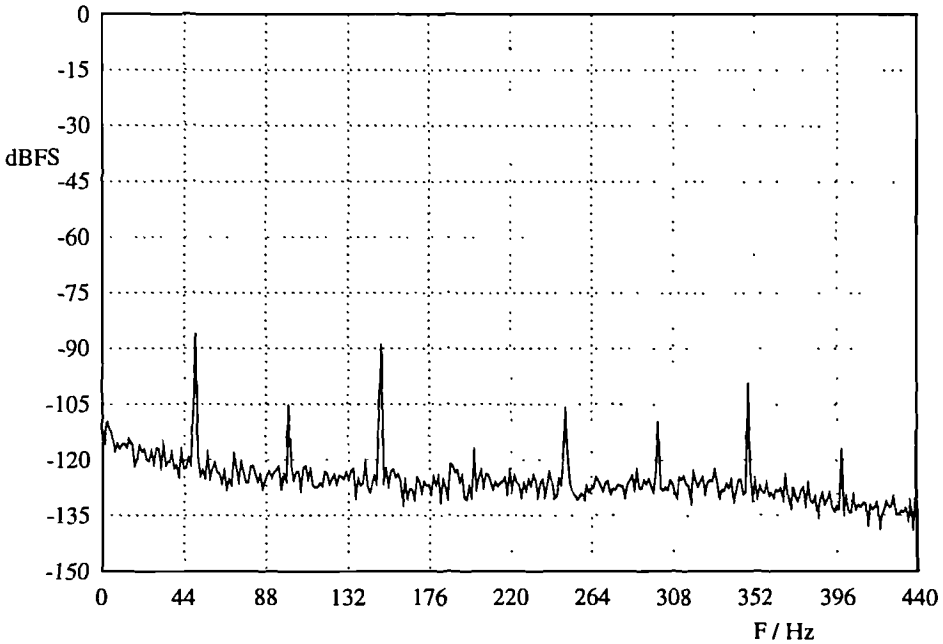


Figure 6.3.5.b : Supply Ripple Leakage Into The Output Signal With Single Linear Regulation.
(nb. : the harmonics of 50 Hz, ... especially the odd order terms)

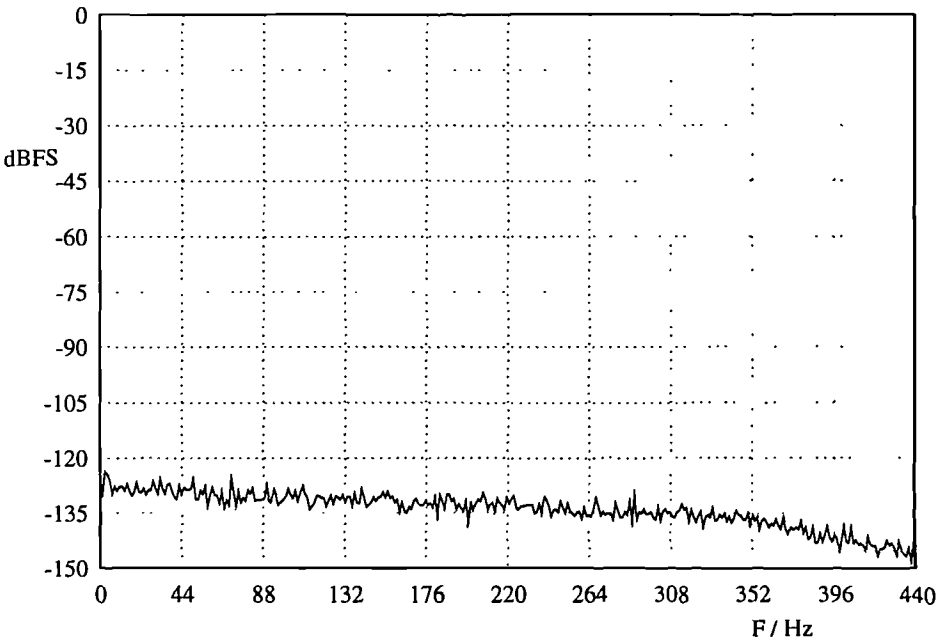


Figure 6.3.5.c : Supply Ripple Removed From The Output Signal With Double Regulation.
(nb. : the 50 Hz term and its harmonics have now been removed)

6.4 : Fast PWMs and Counter Design.

6.4.1 : Overview

Five pulse width modulators were constructed in the course of investigating the best circuit configurations for the PWM, clock and output stages and the best combination of the number of bits, oversampling ratio, modulation depth and modulation type.

Initially, clock multiplication was attempted for a low wordlength symmetric DPWM (8 bit) based on a two counter structure without oversampling. A power switch was included in this design along with a ROM based signal source and an L-C oscillator as a sample rate clock source. This circuit demonstrated that a two counter structure was good but a higher speed design would be necessary to handle more data bits. Also, clock phase control is not a good method of producing high resolution pulses, and clock multiplication is prone to error without low PLL ratios and a very stable source.

The second DPWM design uses clock division to reduce timing errors and was designed to operate synchronously at 144 MHz with up to 12 bit data being scanned from RAM to control symmetric, asymmetric, and single sided PWM types. Data can be programmed by a personal computer or provided (real time) from a TMS32020 DSP IC. No output switch was included but some signal level results of noise shaped and pre-compensated PWM were gained.

Difficulties in matching the propagation delay in this DPWM left 'dead zones' in the clock frequency range at which the board no longer worked due to race hazards. Although the board clearly demonstrated that circuits could be used at such high counting speeds, it also showed that variable clock speed dramatically increases the complexity of the design since race hazards can no longer be avoided by careful timing. Spectral measurements showed that cycling stored data yields harmonically related quantisation noise that gives misleading results if not taken into account. In light of the difficulties encountered, further research was put into counter structures for fast operation, leading to a preference for an asynchronous design.

The third DPWM retained the dual counter structure of both previous designs, but in an asynchronous configuration; dual ports were included to allow full control of both delay and width of each pulse and hence facilitate all modulation types. 180 MHz operation was achieved but clock phase control was avoided to reduce edge errors. Clock division was retained and configured so that operation from a CD player source could be performed. Many tests were carried out at 8x oversampling, using 8 data bits, a second order sinusoidal noise shaper ASIC and 12% guard bands. Complementary output drivers reduced drive circuit current fluctuations so much that near CD quality performance was achieved for the first time. This board proved so flexible that the fourth prototype using sequential asynchronous PALs was designed with identical core circuitry.

The fourth DPWM did not include noise shaping as this was separated as a DSP task knowing from simulation results that sinusoidal noise shaping would be inadequate at low oversampling ratios. The CMOS output devices were replaced by ECL latches to reduce output switching times. The clock division circuitry was removed anticipating a return to PLL clock multiplication. Simple interfacing for 2SCPWM was added, to assist in finding alternatives for low distortion performance avoiding the computational complexity of pseudo-natural PWM pre-compensation.

The fifth DPWM again retains the best features of the previous circuits, but included further

synchronisation circuitry to drive preceding DSP and interface ICs so that a 'stand alone' DAC, not dependent on the source for clocks, could be assembled. ASRC, 10x interpolation, and enhanced sampling were included in this design, for which the modulator which was made in stereo.

6.4.2 : PWM 1 : PLL Clocked 8 Bit Modulator.

In this first prototype DPWM design, a clock source, a data source and an output switch were included besides the basic DPWM unit itself as shown in figure 6.4.2.a . These elements could each be designed in different ways or even replaced with alternatives blocks but they are required in some form to enable proper testing of the DPWM itself. Each will be discussed in brief before the DPWM itself.

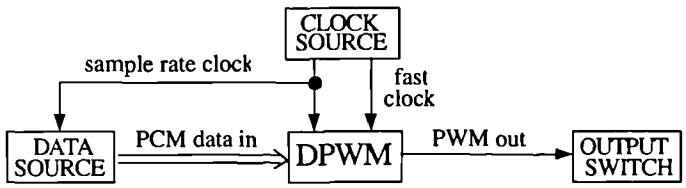


Figure 6.4.2.a : Functional Partitioning Of The First DPWM Prototype

The clock source was based on a high precision R-C oscillator to produce the sample rate clock at 44.1 kHz (74HC4047). From this the fast clock was produced by frequency multiplication using a phase locked loop and division in its feedback loop (PLL : 74HC4046a). 8 bit modulation was to be used in the DPWM so a division by 256 was used in the feedback (74HC393) making the fast clock 11.2896 MHz. Faster PLL operation could not be achieved in the fast CMOS series so special techniques were introduced in the DPWM to accommodate true symmetric modulation.

Knowing that the input clock rate was fixed by the local oscillator the voltage controlled oscillator (VCO) in the PLL was set for a small range (11.0 MHz to 11.6 MHz) reducing the phase error by increasing the loop gain. To ensure zero phase offset once the PLL was 'locked' onto the sample rate clock, a type II edge-sensitive lead-lag phase detector was used despite the worse noise performance compared to the type I [HOR80]. Shielding was included near this circuit to reduce external noise upsetting the loop. Slow response time was introduced by using a second order feedback filter with a mixed proportional-integral control (a zero of 3 second period and a pole at 20 Hz). This ensures that tracking of the sample rate is inaudible and with a small error; from these parameters component values were calculated.

Signal data for test purposes was stored in read only memory (ROM). 4 K Bytes were used to store sinusoid waveforms, with new addresses being generated at the sample rate by a 12 bit counter. Typically this allows 100 cycles before repetition (eg. for a 1076.66 kHz signal) so quantisation in the signal is harmonic and indistinguishable from the modulation non-linearity.

The output switch used in this prototype was a complementary half-bridge based on the IRF510 and IRF9520. By using different die sized MOSFETs (-10 and -20 series) the on-resistance was matched (540mΩ and 600mΩ) but this made the drive requirements different because the gate charge is three times higher in the p-channel device (5.2 nC for the n-channel, 16 nC for the p-channel). Parallel

CMOS buffering was used to drive high speed optical isolators via speed up capacitance. These in turn drive DMOS gate drive circuits. The gate circuits were forced to operate at the verge of shoot-through by increasing the driver circuits supply voltage to the sum of the threshold voltages (about 6-7 volts). 20 ns. edge times were achieved by this circuit configuration but hysteresis in the optical isolation and MOSFETs was seen to introduce distortion which should be avoided in high resolution switches.

The DPWM was designed to produce symmetric modulation by using delay counting from the previous sampling instant and width counting once the delay has been completed. True symmetric performance was attempted by using a 50% duty cycle in the counter clocks and switching between the active high clock and an inverted version of it controlled by the LSB. To ensure these clocks were in close anti-phase to each other, logical EXOR gates were used, setting the unused input high or low to effect inversion or buffering with the same delay.

The data value for each delay is loaded in the second half of the previous sample period since at this time the previous delay counting must have finished in a symmetric modulation scheme. The data for the width count is loaded while the output is low. In this way, asynchronous parallel loading can be used to prevent spurious output signals being sent at inappropriate parts of the cycle but this also requires that data must be valid from just before to just after the delay counting is in operation. To meet this requirement, new data is latched at the pulse centres (the sampling instant). Since the clock phase is determined from the data LSB and could change when the new data is loaded, the LSB used to define the clock phase has to be separately latched so that it is held constant throughout both counters' operating times. This was achieved by using the falling edge of the output pulse to latch the LSB value for the next cycle, but this does mean that absolute 100% modulation should be avoided as the LSB would then no longer be updated. Figure 6.4.2.b (below) shows the timing cycles used to ensure the high speed loading and counting are kept separate.

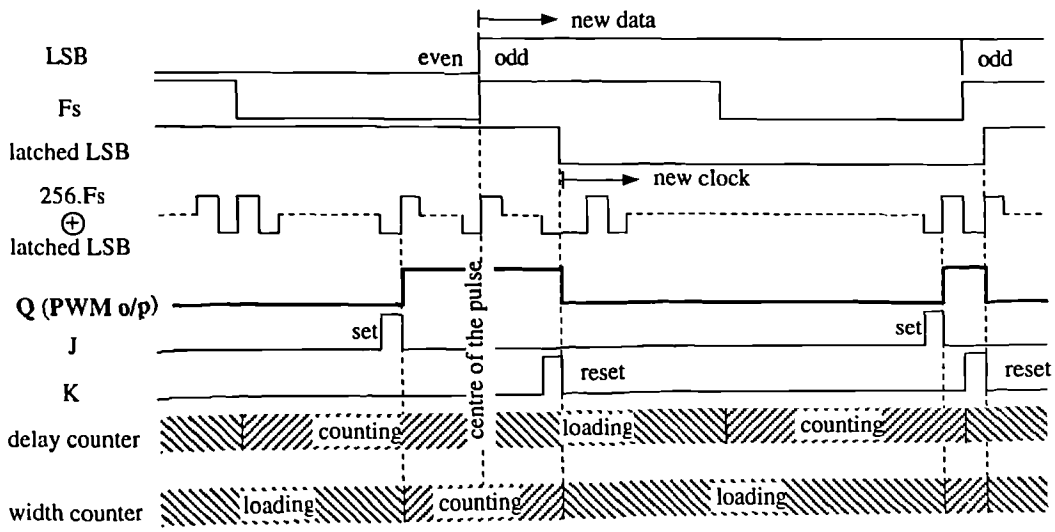


Figure 6.4.2.b : Counting Schedule For Phase Controlled Symmetric Modulation.

Delay periods are calculated from the data value by shifting the data bus priority when applied to the delay counter (hence losing the LSB). Thus only half the remaining time in the sample rate period is used as the delay. Complementary counting (counting up to full instead of counting down to

empty) was used to effect the counting of remaining time rather than elapsed time. A schematic diagram of this way of controlling symmetric modulation is shown below.

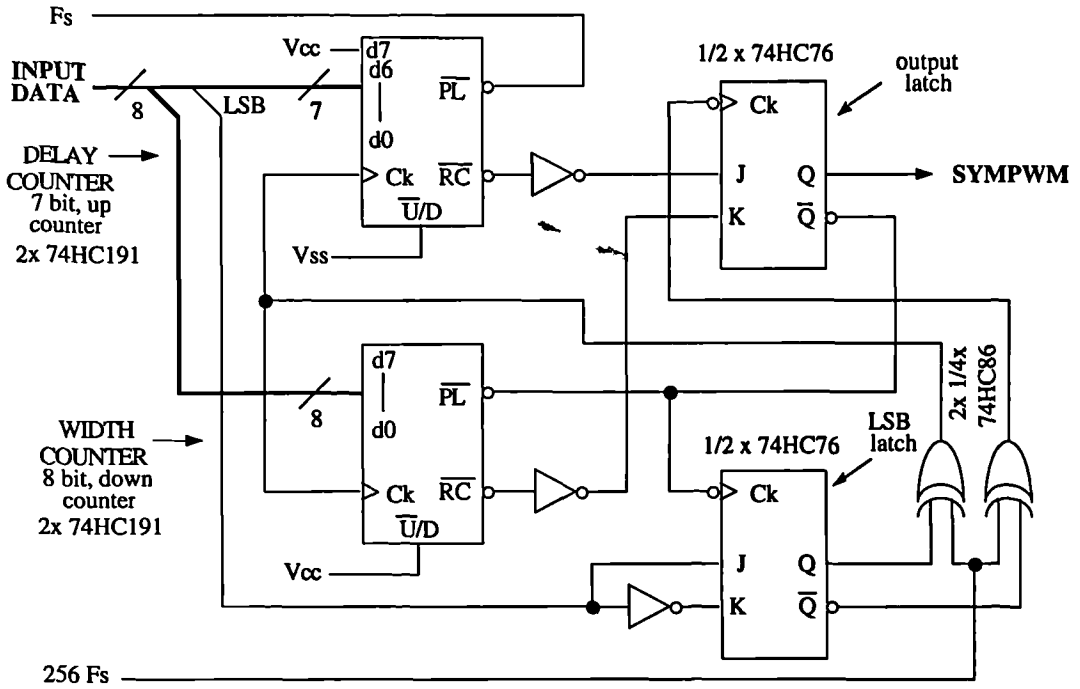


Figure 6.4.2.c : Symmetrically Modulated DPWM Using Phase Controlled Clocking.

Shown below are oscilloscope photographs of the output latch operation, the output buffer operation and the output current and voltage waveforms for a 1.1 kHz sinusoid.

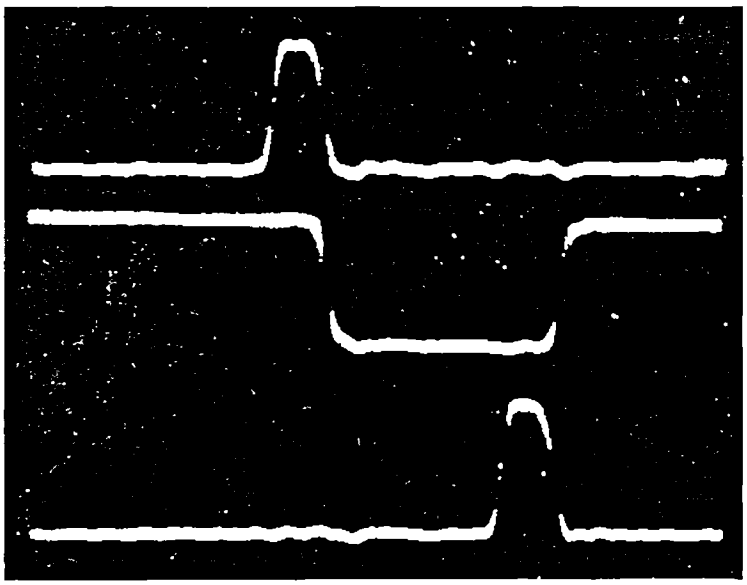


Figure 6.4.2.d : High Speed 'SET' And 'RESET' Signals To The Output Latch

Top Trace	: J input to the output latch,
Middle Trace	: Not-Q output of the latch,
Bottom Trace	: K input to the output latch.
Voltage scales	: 3 v/cm all traces, Timebase 50 ns/cm.
Data value	: 2, Pulse Width : 176 ns.

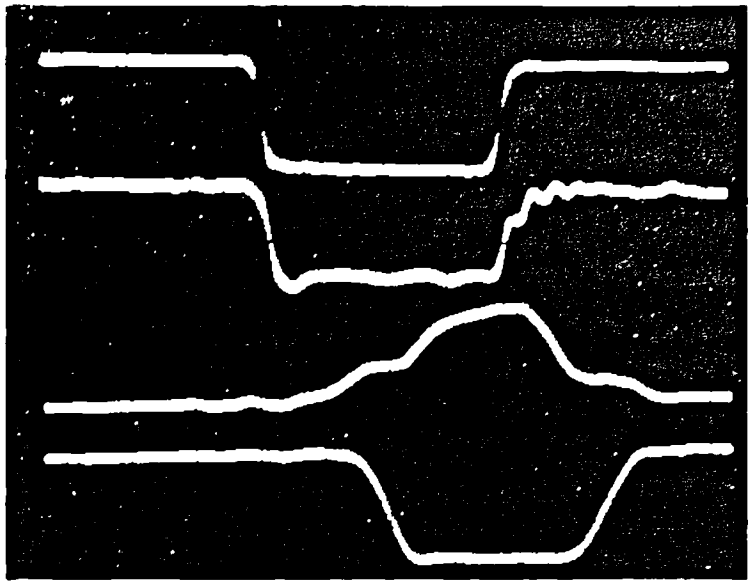


Figure 6.4.2.e ; Output Latch, Buffered Output, Opto-coupler Output And MOSFET Drive Pulses.

Top Trace	: Not-Q, output latch
Next Trace	: LED drive voltage,
Next Trace	: Opto-coupler response,
Bottom Trace	: Gate drive to the power MOSFETs.
Voltage scales	: 3 v/cm all traces, Timebase 50 ns/cm.
Data value	: 2, Pulse Width : 176 ns.

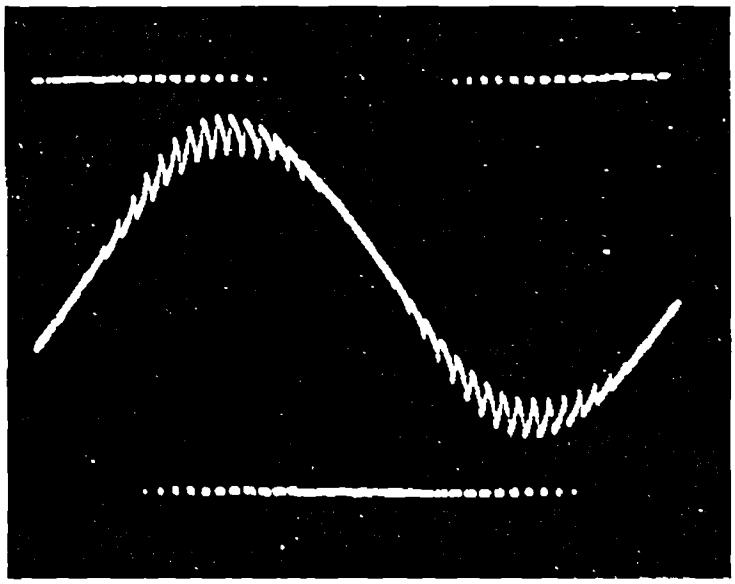


Figure 6.4.2.f ; Output Load Current And Voltage : Half Bridge Output And An L-C Filtered Load.

Note: in this output measurement the voltage can be seen leading the current substantially. This load is a 15.9 kHz (- 3 dB. point) low pass filtered loudspeaker which appears at 1076 Hz as 7.96 + j 6.76 Ω. Full schematics and layout diagrams for this prototype can be found in appendix A6.

6.4.3 : PWM 2 : Divided Clock 12 Bit Modulator.

Recognising the need for higher speed modulator clocks in the first prototype, a design was developed for operation at as high a speed as possible with advanced CMOS in place of high speed CMOS. Knowing that an output synchronous with the clock was required, synchronous counters were compared for the fastest carry speed possible. The 74AC169 4 bit synchronous counter was chosen as the fastest, counting at typical frequencies of 154 MHz. Problems arose with this device later because at top speed, the end of count indication is not in phase with the clock cycle from which it was decoded. For a fixed clock frequency this can be compensated for but this severely restricted the versatility of the final circuit when variable clock frequencies were required. A basic block diagram of this second prototype is shown below.

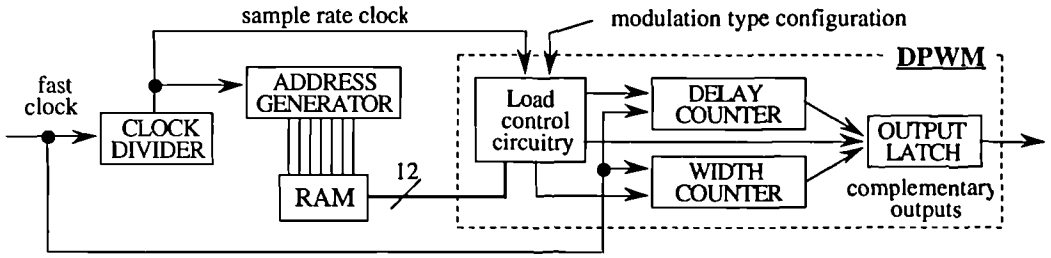


Figure 6.4.3.a : Basic Block Diagram of the Second DPWM Prototype.

As before, memory was used for storing signal waveforms but the opportunity to use less than the full memory was designed into the addressing of the data. This allows considerably more flexibility than the first prototype and by including a port on the data bus with latching, real time signals could be fed in when the memory ICs were removed.

To avoid PLL frequency multiplication with its associated errors, clock division was included on board using division circuits based on the same counters as the PWM itself. The divider was made programmable so that various integer factors (between 3 and 4095) of the fast clock could be used as the sample rate. Slow end of count detection from the 74AC169 adds one cycle to the programmed number of cycles and this should be taken into account when programming the ratio.

With 12 bit counters used in the delay and width counters, additional logic was added to preset the unused bits when lower wordlengths were in use. Only the delay counter has to be preset in this way since unused bits in the width counter do not affect the count down process (zero added in the MSBs does not affect a down counter but does interfere with an up-counter's operation).

Since the end of count has to be set up before the counting begins on the PWM counters and the propagation delay of the 74AC169 was known to be particularly long, the MSB counters (bits 4 to 11) are set up one cycle early allowing immediate response from the counter once enabled. A careful combination of synchronous and asynchronous control was used for this since the response time for the asynchronous controls of the end of count latches was faster than the data inputs (74AC109).

Extreme care was taken to design the layout for this PCB so that it could handle 150 MHz operation. Surface mount devices for lower lead inductance, short track lengths, double power planes and heavy decoupling were all used to try to maintain as high a working clock rate as possible. Crosstalk was examined in clock lines and asynchronous controls and minimised by appropriate

geometry and ICs were grouped not only according to minimum track lengths but also by speed to keep the supplies as 'clean' as possible. Transmission strip-lines are used for the highest speed connections.

Rise times of 3 ns. were achieved in the PWM output pulses but for clock speeds more than 10% different from the designed speed, race hazards could no longer be removed by the built in timing and the PWM's operation failed. Alterations were made to allow operation at 80 MHz with a proviso that 100 % modulation depth should be avoided. An interface to a PC was constructed for programming the RAM and binary searched data was down-loaded for spectral measurement of uniform and pseudo-natural operation. Software was written to prepare output files; example results from these with 16 bit, 1 kHz tone after second order noise shaping are shown below. A 176 kHz carrier was used in both cases (ie. 4x oversampling). A full schematic along with a layout photograph is included in appendix A6.

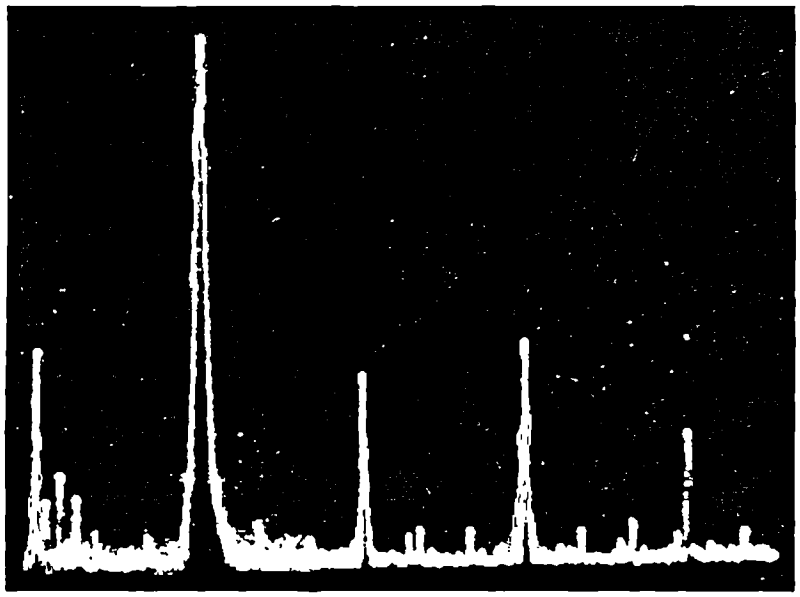


Figure 6.4.3.b : Symmetric Modulation @ 98% Modulation Depth (- 88 dB. floor, 0-5 kHz)

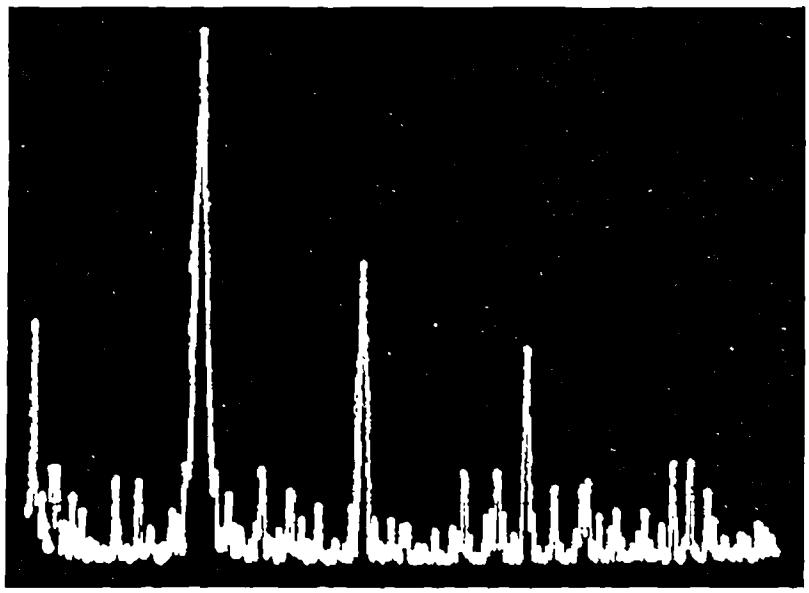


Figure 6.4.3.c : Trailing Edged Modulation @ 98% Modulation Depth (- 82 dB. floor, 0-5 kHz)

6.4.4 : ASIC Fast Counter Designs.

Since working with the high speed synchronous designs had proved so difficult, counter structures for high speed operation were investigated. Synchronous and asynchronous designs were compared with a novel non-binary counter based on a maximal length shift register. For fairness, a rising edge triggered J-K flip-flop was designed from individual gates to form the basis of each of the counter structures. Once each top level design has been completed, fan-out capabilities were manually checked to ensure that it was the structures that were being compared, not the limited drive capability of one part of the circuit. This design is shown below.

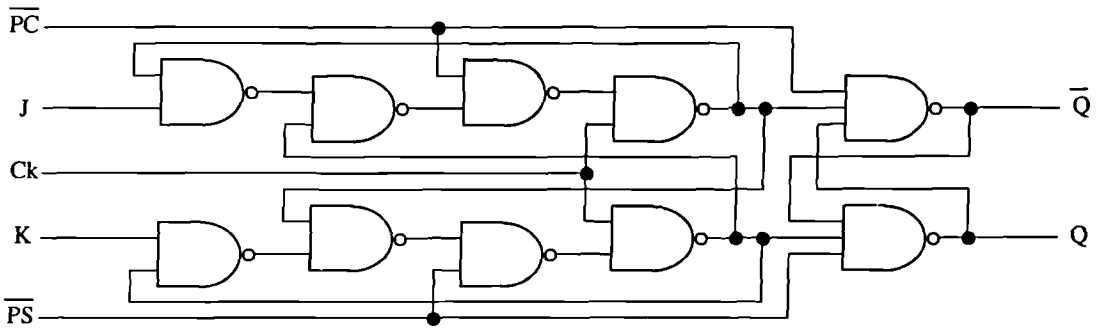


Figure 6.4.4.a : J-K flip-flop For Fast Counter Structure Tests

Initially the asynchronous counter was not considered the best choice of circuit since use of a ripple carry causes the end state to suffer race hazards when multi-bit counters are being used. This forces a reduction in the counting rate well below the maximum toggle rate. A circuit segment for this type of counter is shown below.

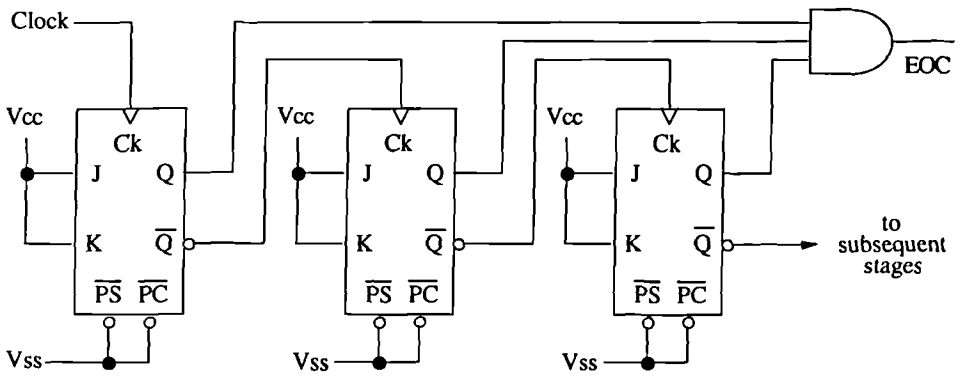


Figure 6.4.4.b : Asynchronous Counter Structure

Although in simulation (using 'SOLO') the toggle rate for the asynchronous circuit was measured to exceed 250 MHz, when the output are logically NAND-ed to detect when all zeros are present (ie. a down counter end point) the system could only be operated at 77 MHz (7 bit case).

Two circuit configurations were considered for the synchronous multi-bit counter design. One with each flip-flop's state dependent on its previous state and the 'carry' of less significant bits, and the other with each flip-flop dependent on its previous state and all the less significant bits. In the first of

these two, the carry ripples much as in the asynchronous design, leading to a counting speed limit which decreases for large wordlength counters. A circuit segment for a down counter is shown below.

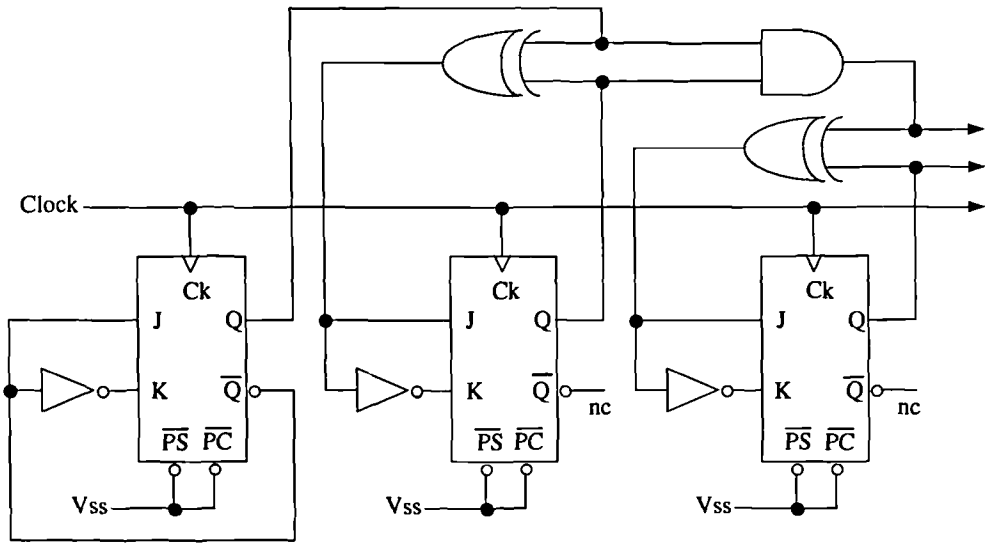


Figure 6.4.4.c : Ripple Carry Synchronous Counter

In the second type of synchronous counter, the look-ahead carry circuitry becomes increasingly more complex as the wordlength of the counter increases leading to fan-out problems and gate-width problems. This also leads to a reduction in the counting speeds possible for large wordlength counters. A circuit segment of this configuration is shown below.

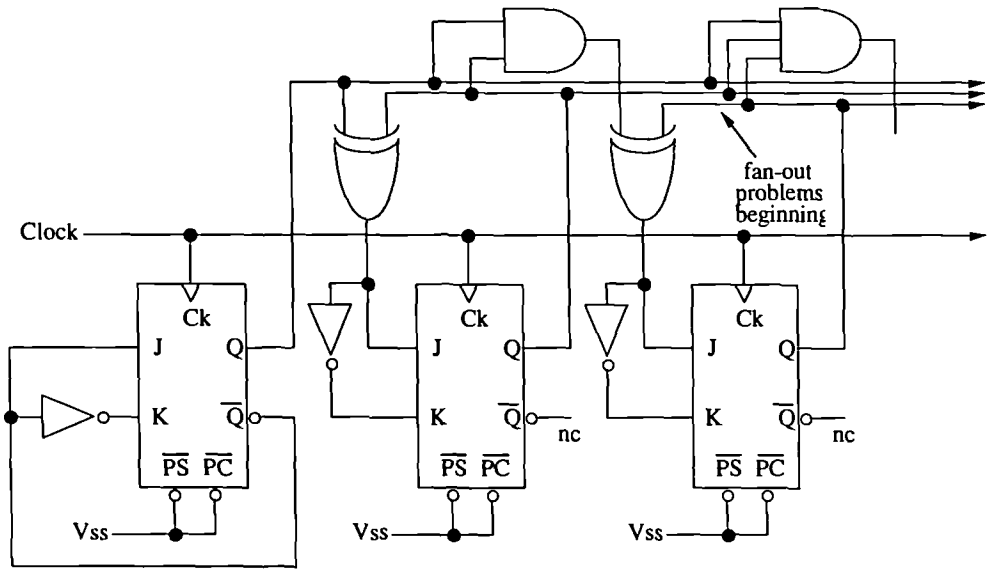


Figure 6.4.4.d : Look-ahead Carry Synchronous Counter Structure

From investigating designs for high speed operation without complicated feedforward or feedback networks, a shift register was considered since for low wordlength counters this can provide counting at the toggle rate. The number of counter elements is no longer the logarithm to base two of the counter range but rather the counter range itself. This leads to enormous complexity unless the

design is split appropriately. By using very simple feedback, a maximal length sequence can be obtained from the shift register. This yields a very similar counter range to the conventional binary counters but with one excluded state and represents a good compromise between the complex no-feedback system and the slow simple counter systems. A short example of this structure is shown below.

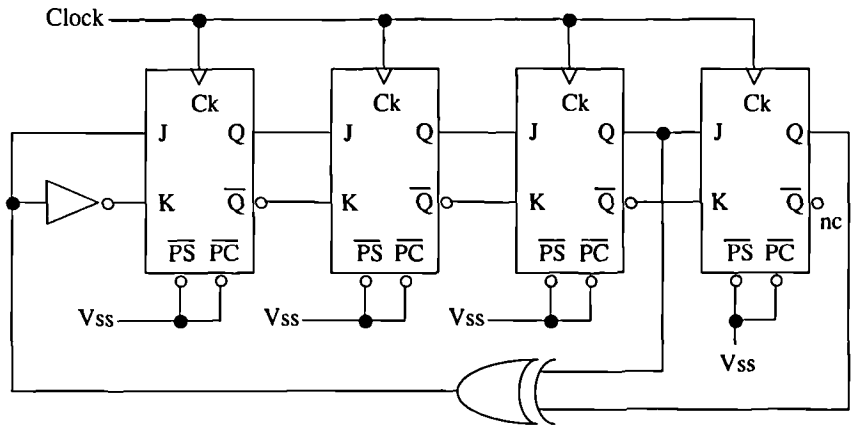


Figure 6.4.4.e : Maximal Length Sequence Shift Register.

The feedback in most maximal length shift registers employs EXOR gates which are slow, so to maximise speed in the feedback network, the EXOR was split into two NOR functions, using latches between the NOR operations to reduce the propagation delay between any two stages. This additional latching forces the feedback taps to be taken from an earlier stage of the shift register so the revised structure is as shown below.

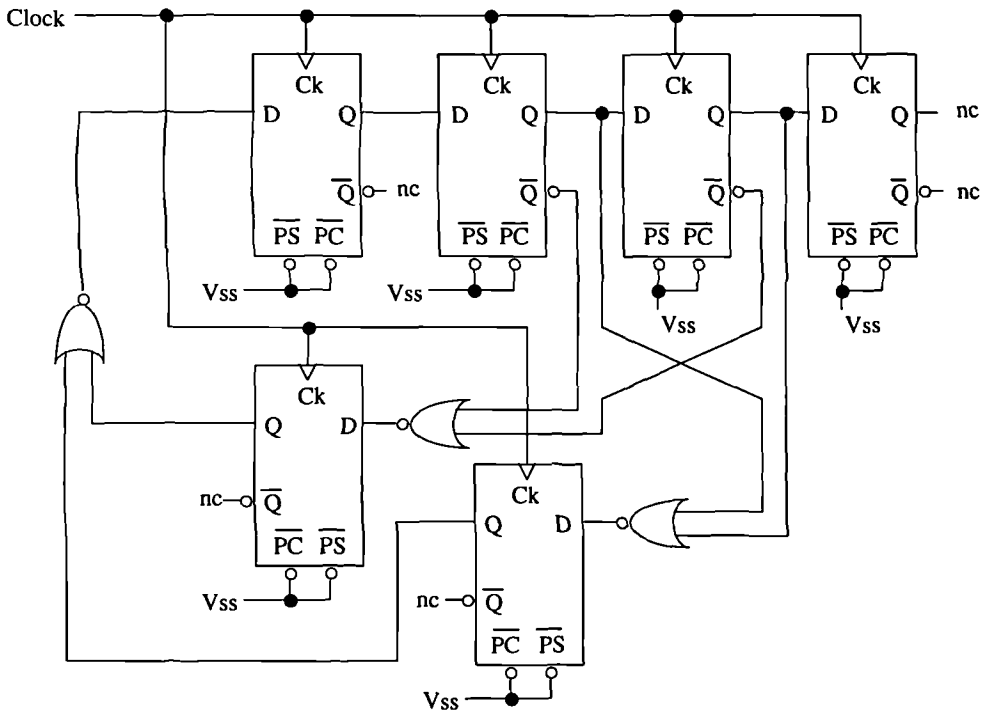


Figure 6.4.4.f : Revised Maximal Length Sequence Shift Register.

Since this structure does not count in binary, input encoding and output decoding is required which may be intricate. Time is available to carry out the input encoding since this can be done in a DPWM while the counter is waiting (loading) but the output must be decoded as quickly as possible. To do this the all 'one' state should be used as the end state in all cases since this is simple to decode, and can be anticipated as it develops along the shift register. Counters based on this system will be referred to as maximal length sequence counters (MLSC).

Using this network, the MLSC was compared to the two synchronous designs for a seven bit case. The two synchronous counter were able to count up to a clock frequency of 100 MHz and 125 MHz respectively, but the MLSC continued to operate at up to 177 MHz confirming that the feedback minimisation was worthwhile.

Noting that the use of simple end states dramatically simplifies the 'end of count' decoding (EOC), and that in the shift register this propagates allowing some anticipation of the output, a similar approach was supplied to the asynchronous structure to take advantage of its inherent speed superiority. In the asynchronous counter, the EOC detection limits the clocking speed because of time skewing in the availability of the outputs. When counting down to zero or up to all ones in a binary counter, the MSBs are completed with first, so the development of the EOC for this case can be done progressively, relying on the propagation delays to provide 'covering logic' for the race hazard that is built in by using progressive EOC detection. This structure is shown below.

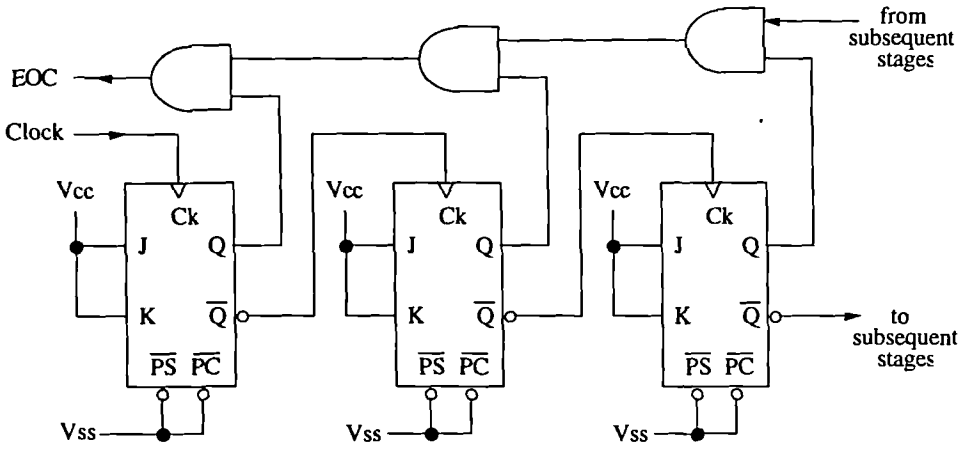


Figure 6.4.4.g : Asynchronous Counter Structure using Progressive EOC Detection

Settling time for the counter must be provided immediately after loading to allow the output to settle, but in the case of the DPWM this can be provided during a guard band interval. Simulation of this circuit proved operation at 250 MHz could be achieved, bettering all the synchronous structures. From this basis, the third PWM prototype was developed.

6.4.5 : ASIC DPWM Design.

An ASIC DPWM design was attempted based on the previous prototypes and the results of comparing fast counter structures. The clock divider, delay counter and width counter were laid out using the structure of figure 6.4.4.g using a simplification of the J-K flip flop to make it toggle in all

instances except when being accessed by the asynchronous inputs 'PRESET' and 'PRECLEAR'. The two counter structure of the initial prototype was retained, but the clock phase selection for symmetric modulation was removed. A block diagram of this structure is shown below.

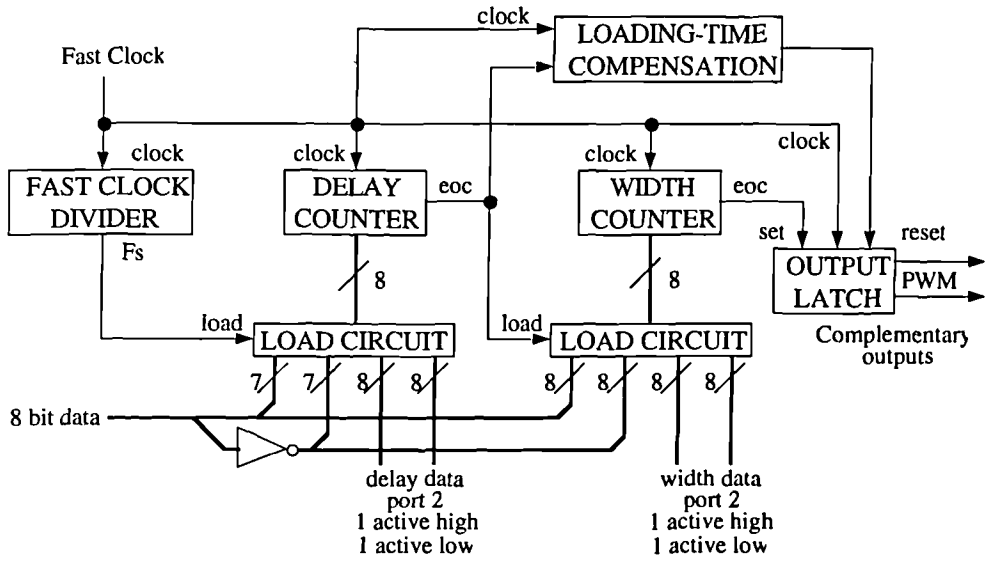


Figure 6.4.5.a : Block Structure Of The proposed ASIC DPWM Prototype

At first the circuit was configured so that the divider EOC caused loading of the delay counter and the delay counter EOC caused loading of the width counter. By adapting the divider to accept different ratios, different modulation depths could then be used for simulation.

Loading logic was prepared using a quadruple 2->1 line multiplexer (74HC158 model) which has a 'clear' input to force all the (active low) outputs into the high state. When the clear input is driven by the relevant EOC signal, data appears on the counter's asynchronous load pins, blocking counting until released. Since the load signal disappears after the initiating edge for the counter it is loading the delay and width counters miss one clock edge and their output duration is one cycle too long. In the case of the delay counter, this represents a time shift and can be ignored. In the case of the width counter the increased period directly corrupts the data so the start of the output pulse has to be similarly delayed to the delay in the counter starting to count. One flip-flop in the 'SET' path from the delay counter to the output latch deals with this.

By using a 2->1 line multiplexer in the data path, not only can the loading of the counters be easily controlled, but direct access to the delay and width counter data can be gained. Thus, through one pair of ports, the appropriate relationship between the delay and width for an asymmetric PWM can be wired-in and through the appropriate use of the other ports, leading or trailing edged modulation can be achieved (by setting one counter to a fixed value). The use of the second ports is arbitrary so 2SCPWM and PNPWM or AOAPWM can be applied making the modulator very versatile without compromising the clock speed that can be used in any way, however both true and invert data has to be applied since both the PRESETs and PRECLEARs of each counter have to be controlled. It is by putting the data inversion before the load circuit, and doubling up the load circuit itself, that ensures minimum delay between the loading and counting.

The 74HC158 clear input is in fact active low, so the last stage in the EOC detect circuit which drives this input is NAND instead of AND. The 74HC158 was chosen specially for this feature since the drive capacity from the NAND gate is higher for the same propagation delay; hence this last element in the EOC detect circuit has the capability to drive both the PRESET and PRECLEAR associated loading circuits. The output from the width counter does not drive any such load, so its polarity can be kept 'true', and the J (active high) input of the output latch is associated with it. The output latch was chosen to be the 74HC109 whose K input is active low, matching the output from the delay counter EOC which was active low to gain the high drive capability of the NAND function. Schematics for this ASIC DPWM are to be found in appendix A.5.

6.4.6 : PWM 3 : Oversampled 8 Bit Modulator With Noise Shaping.

Having modified the ASIC DPWM to use 74HC models throughout, the circuit was known to also work if implemented in discrete logic rather than ASIC. Since further modifications might be made to the DPWM at a later date, production of the ASIC was shelved to finalise a design in discretes.

Software simulation of the DPWM was suggesting at this stage that 8x oversampling was likely to be the best sample rate as a balance between low output pulse repetition rate and a high enough oversampling to allow noise shaping that did not re-introduce noise through the PWM's non-linear behaviour.

With the new objective of attaching the PWM DAC to a CD player source, an 8x oversampling CD player was adapted to provide serial data and accept being driven by an external clock (see section 6.3.1). This added a constraint that the fast clock (previously simulated at 90.3168 MHz) has to have an integer ratio with the clock fed to the CD player (16.9344 MHz). The lowest integer ratio to allow 8 bit operation and 8x oversampling is 6, so the fast clock was revised to operate at 101.6064 MHz. Division by six was designed with a 50% output duty cycle, and a single clock period output to use in the EOC circuit (see section 6.3.3).

Driving the DPWM at 101.6064 MHz and supplying data at 352.8 kHz, allows 288 fast clock periods per output pulse. Pre-scaling of the data to use this range would destroy the effects of noise shaping or require revised circuitry right through the system. Since high modulation depths were known to cause distortion in the output stage if not the PWM itself (from the second prototype), guard bands of 32 cycles were added to the design, during which loading of the counters was achieved. This not only allows slow loading of the counters, but can also provide settling time for the load circuits in which race hazards in any EOC logic can be covered. For DC symmetry, 16 cycles were added to the high portion of the pulse and 16 cycles were added to the low portion of the pulse. The guard band circuits and the loading circuits for the various different types of modulation are discussed in detail in section 6.5 .

For high resolution operation, the second order sinusoidal noise shaping ASIC was included on the PCB (see section 6.1) and configured with its own loading PAL (see section 6.2.5). This not only allows moderate noise shaping to be provided on-board, but when appropriately configured, the noise shaper can be bypassed allowing external (floating point) noise shaping. By removing this device

and replacing it with the multistage noise shaper (see section 6.2), up to fourth order noise shaping can be provided on-board.

The overall block diagram of this PCB is shown below.

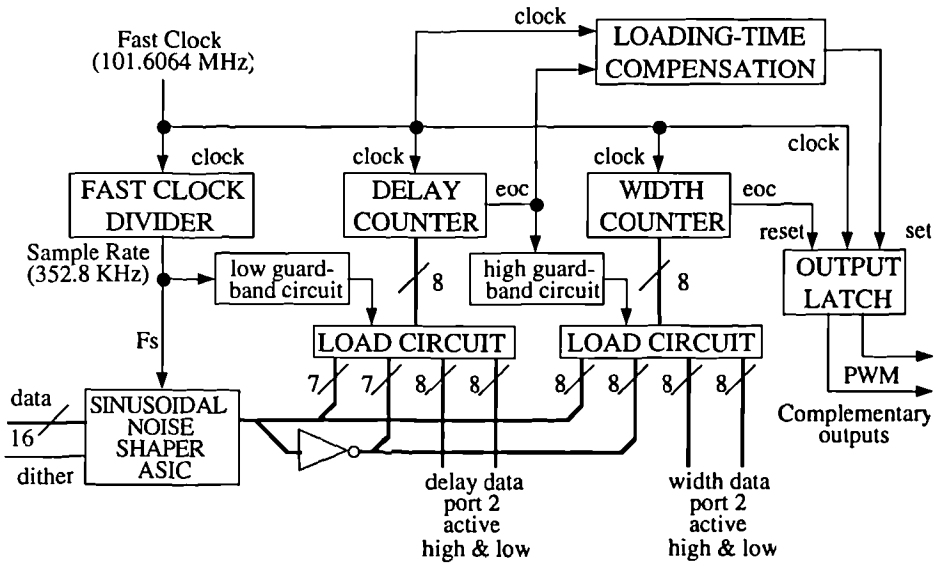


Figure 6.4.6.a : PWM III Prototype Block Diagram

Spectral results were taken for sinusoid tones and twin tones using trailing edged, lagging asymmetric and alternating odd asymmetric AD PWM and matched theory very closely. Conducted noise was seen to contaminate the theoretical SNR, so double supply regulation was added as described in section 6.3.5; also , induced interference was picked up from local sources (PC monitors, and switched mode PSUs) so careful screening and separation was used in noise-critical tests. Broadband tests showed the interpolator in the 8x oversampled CD player allowed spectral replicates to reach its output that were only some 40 dB. lower than the signal; this was replaced as described in section 6.3.4. Digital switching noise on the 5v rail was observed in the output, so second output latching was provided with complementary operation to force constant current operation at audio frequencies. This was also supplied with its own regulation which altogether improved the noise floor from circa -105 dBFS to near -120 dBFS. Output switching was added in a complementary half bridge configuration to provide up to 15 W RMS audio signal power. Measurements of sinusoids at -15 dBFS (2 W RMS output) gave THD at 0.0035% from the switch after notching out the fundamental. This demonstrates that class D output can work for power amplification purposes. Better switches and switching topologies are required for better quality and higher power; these are discussed in chapter 7.

A sequence of results after each element in the PWM DAC chain is shown overleaf (in this sequence the original interpolator was used to demonstrate its poor spectral performance). The spectral performance for various input tones and twin tones is shown on the following pages. Hardware and simulated performance is compared on the third page. These demonstrate that the harmonic performance of the hardware very closely matches that simulated in the design stages. Also the available SNR can be seen to be very high in all cases, suggesting that the last problem to solve for a high quality DAC, is pre-compensation of the data to eliminate as much of the harmonic distortion as possible.

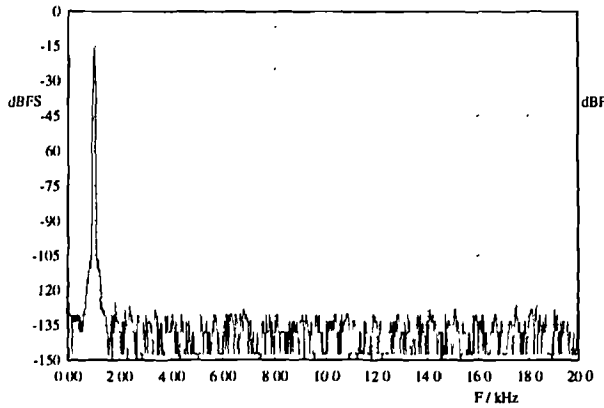


Figure 6.4.6.b ; Input Signal
(Fs=44.1 kHz, signal : 1 kHz @ -15 dBFS)

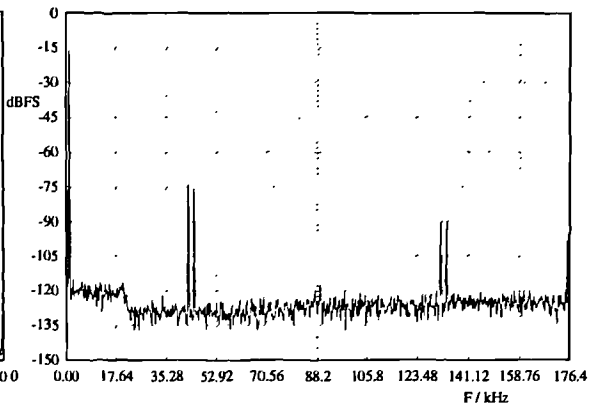


Figure 6.4.6.c : 8x Interpolated Signal.
(Fs=352.8 kHz, nb. replicates & noise floor)

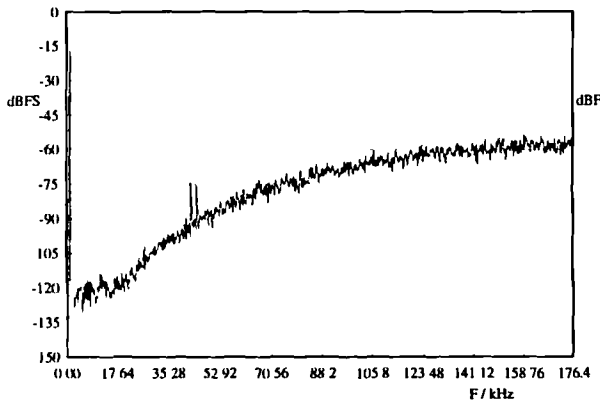


Figure 6.4.6.d ; Broadband Noise Shaped Signal
(nb. high noise power gain & spectral replicates)

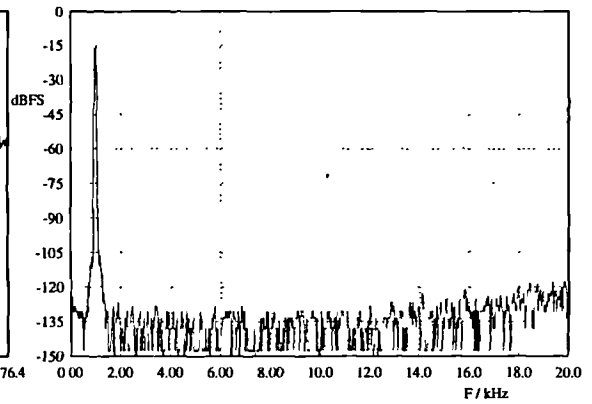


Figure 6.4.6.e : Audio Band Noise Shaped Signal
(nb. linear performance, slight noise floor incline)

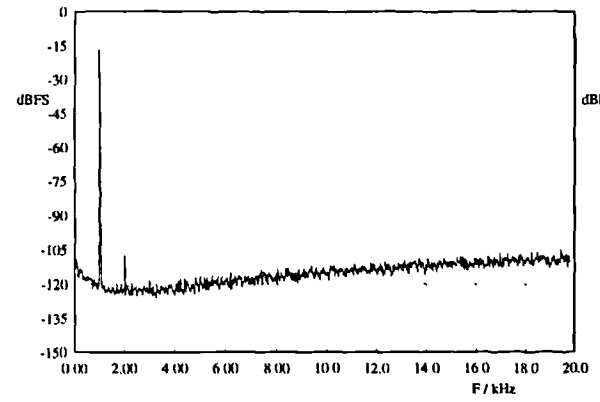


Figure 6.4.6.f ; Audio Band LAPWM Signal
(analogue filtered, nb. PWM's harmonic distortion)

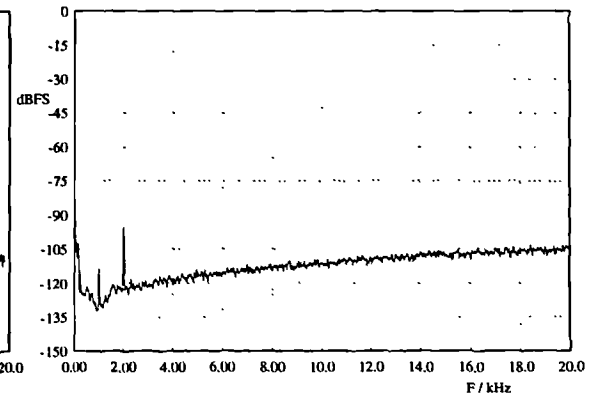


Figure 6.4.6.g : Audio Band Amplified Signal.
(notched fundamental, nb. additional harmonics)

Spectra 6.4.6.h-m are all measured from 8 bit LAPWM with fourth order sinusoidal noise shaping

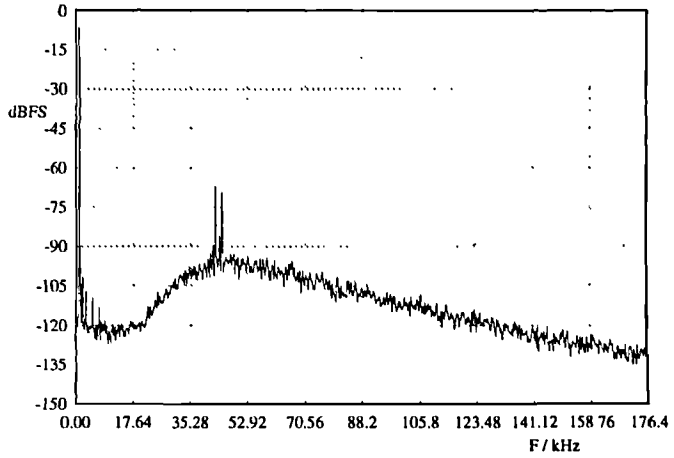


Figure 6.4.6.h : Broadband Filtered Output
(nb. spectral replicates and filtered floor)

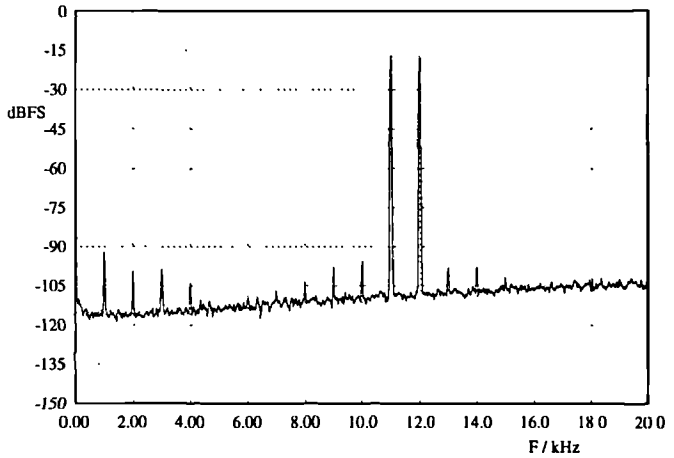


Figure 6.4.6.i : Twin Tone Performance.
(signal : 10 & 11 kHz, IMD @ -93 dBFS)

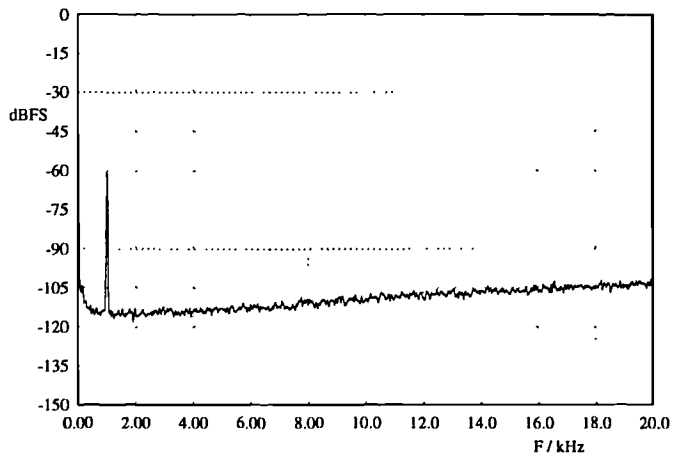


Figure 6.4.6.j : Low Amplitude Performance
(signal : 1 kHz @ -60 dBFS)

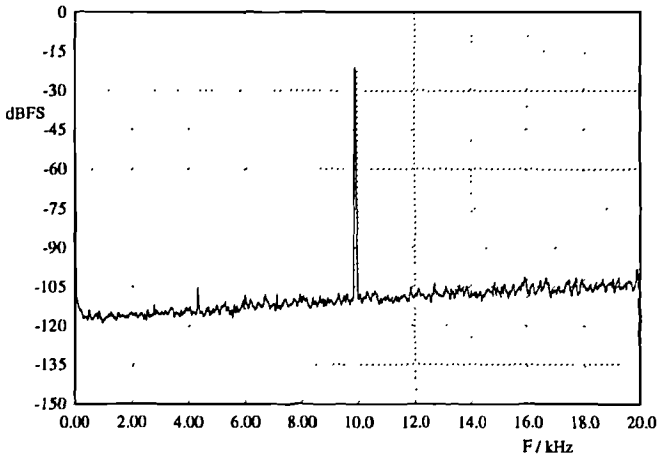


Figure 6.4.6.k : High Frequency Performance.
(i/p: 10 kHz, -20 dBFS, 2 nd harmonic @ -98 dBFS)

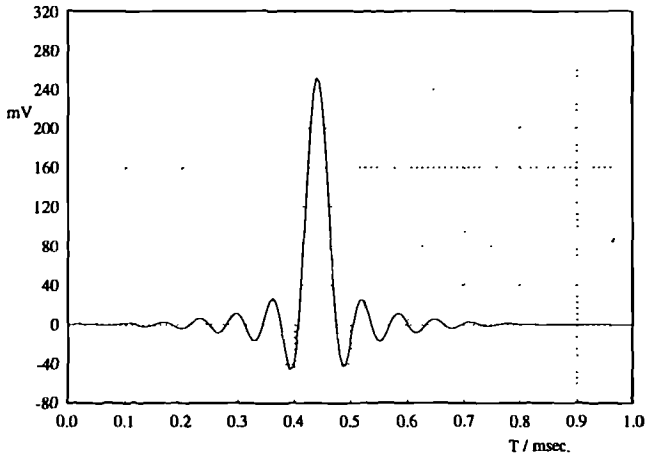


Figure 6.4.6.l : Impulse Response
(nb. this is dominated by the first interpolator stage)

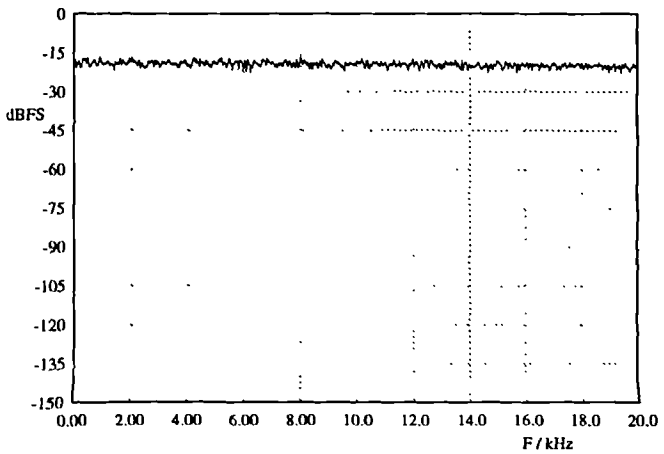


Figure 6.4.6.m : White Noise Response.
(nb. passband roll-off because of output filtering)

Spectra 6.4.6.n-s compare hardware (left) with simulation (right). All use LAPWM & -6 dBFS i/p

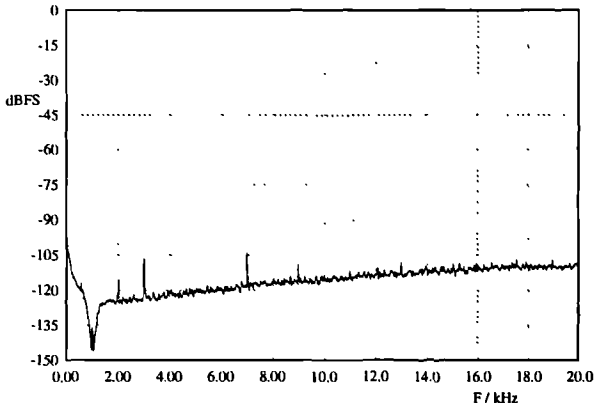


Figure 6.4.6.n : Measured Output (notched fund'tl)
(signal : 1001 Hz, THD @ -97 dBFS)

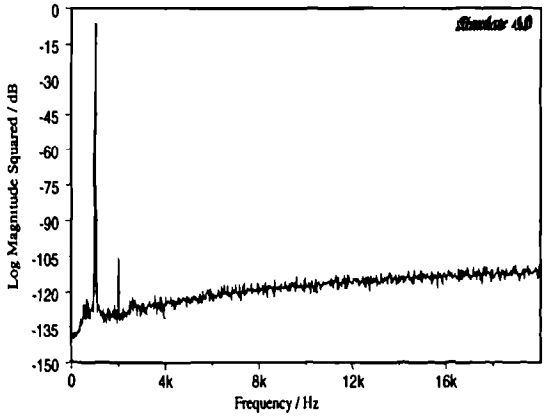


Figure 6.4.6.o : Simulated Output
(signal : 1001 Hz, THD @ -106 dBFS)

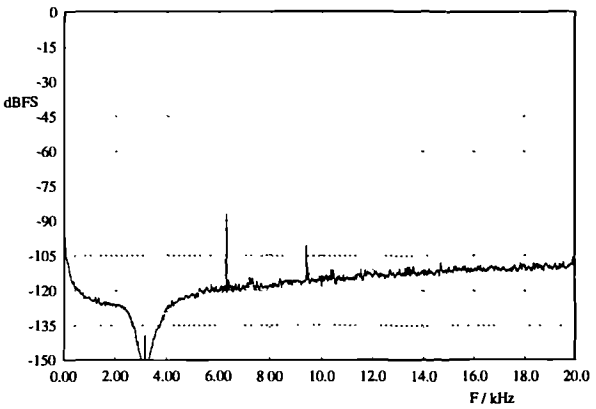


Figure 6.4.6.p : Measured Output (notched fund'tl)
(signal : 3149 Hz, THD @ -86 dBFS)

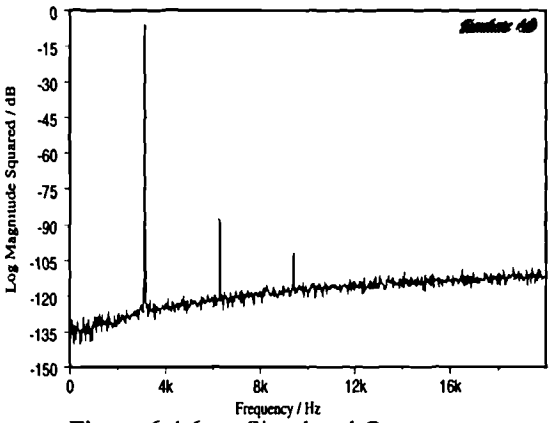


Figure 6.4.6.q : Simulated Output
(signal : 3149 Hz, THD @ -86 dBFS)

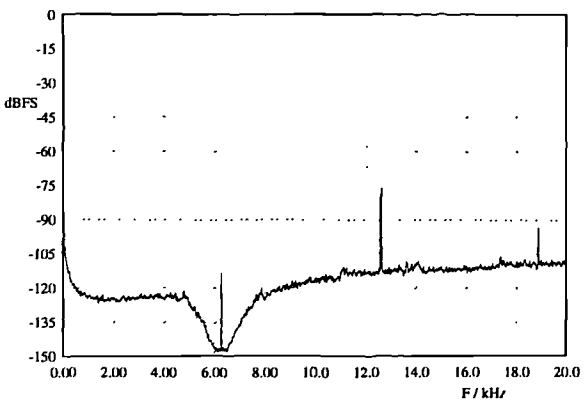


Figure 6.4.6.r : Measured Output (notched fund'tl)
(signal : 6301 Hz, THD @ -76 dBFS)

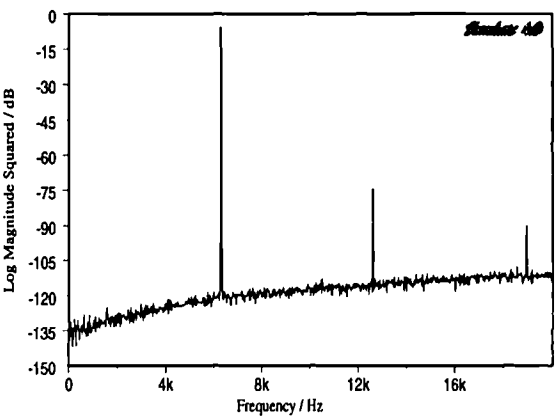


Figure 6.4.6.s : Simulated Output.
(signal : 6301 Hz, THD @ -76 dBFS)

6.4.7 : PWM 4 : External Clock, PAL Modulator.

Sine only a kernel of the DPWM III prototype needs to be operated at the fast clock rate, much of the complexity of that prototype can be assigned to slow circuits (and in particular PAL), making a more re-configurable design. The number of bits in the modulator can then be reprogrammed quickly along with the modulation index used instead of rewiring or re-laying out another full design.

The high speed kernel (>20 MHz) involves the output latch and resynchroniser, the first two bits of the delay and width counters, the load time pre-compensation and the fastest parts of the guard band control circuits. These can all be accommodated in seven 74AC109 devices. If the fast clock and sample rate clock can be provided off-board no counting frequency divider or PLL frequency multiplier is required so only the width counter, delay counter and guard band counters are required. These can be accommodated within three PALs. This could theoretically reduce the circuit size to just ten devices plus some patch logic for the high speed loading of the kernel's data.

With a reduced design, space on a single 'euro-card' becomes available for adding the circuitry for 2SCPWM so that enhanced sampling can be taken advantage of. This dramatically reduces the amount of audible distortion compared to single sample per pulse uniform modulation types. In the future this may provide an alternative modulation type to PNPWM with an acceptable compromise between the (expensive) floating point pre-compensation and 2SCPWM's slightly worse harmonic and intermodulation performance. Since power switching is known to be very difficult to design this may mask any advantages drawn from the use of more advanced (and more linear) pre-compensation techniques.

To reduce jitter at the output yet further, ECL output resynchronisation was used in this design. The asynchronous PAL counters and MSB load logic was programmed into 20RA10 devices, and the data preparation for 2SCPWM was performed by discrete adders and multiplexers (see section 6.5) . A top level block diagram for this circuit is shown below but full schematics for this prototype are to be found in appendix A6 along with a layout photograph.

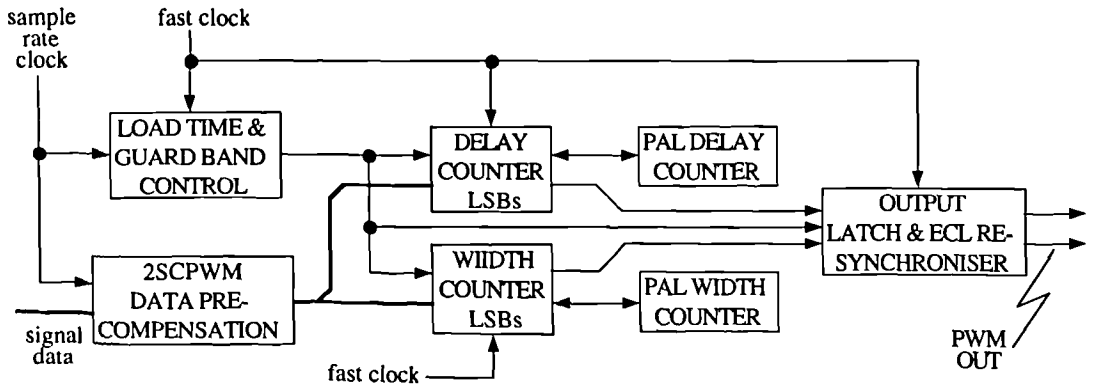


Figure 6.4.7.a : Block Diagram of DPWM Prototype IV

6.4.8 : PWM 5 : Stereo 2SCPWM Modulator With ASRC Input Synchronisation.

The fifth design in this chronology of circuit development for PWM based DACs was designed with commercial implementation in mind. The previous circuits were examined for their best and worst features, leading to the following recommendations :

1) a two sided modulation type should be used for its low distortion performance,
2) a high precision (optimised) noise shaper should be used rather than a sinusoidal one,
3) a low pulse repetition rate (PRR) should be used to simplify power stages,
4) simple pre-compensation should be used to keep distortion down,
5) a local oscillator should be used for the fast clocks with input synchronisation by ASRC,

and, 6) double power supply regulation and output re-synchronisation should be used.

These features were designed into a final system, a block diagram of which is shown below, before a brief explanation of the reasoning behind the choice of parameters and functional elements.

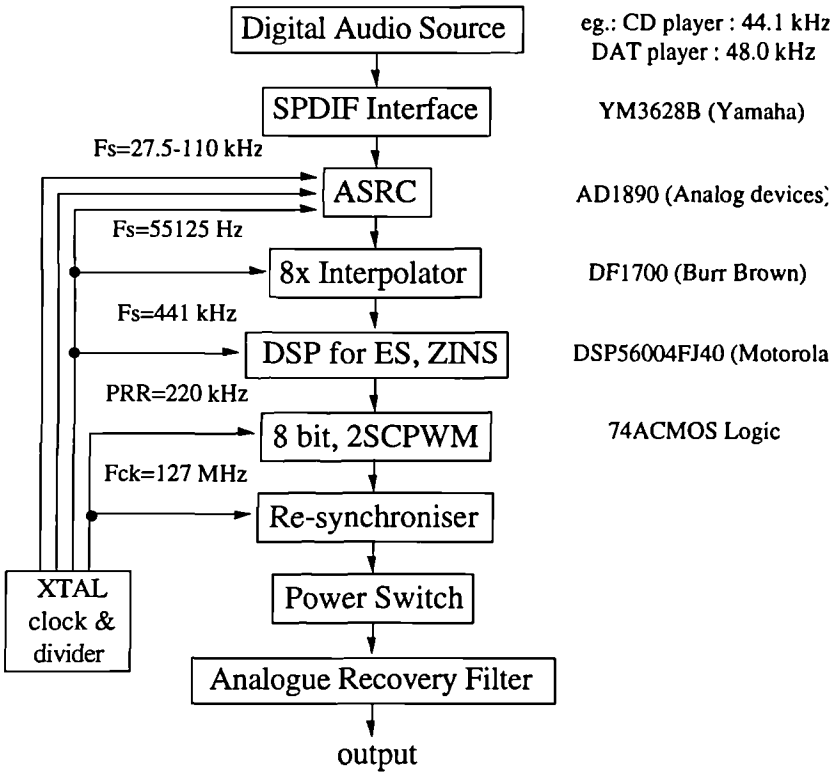


Figure 6.4.8.a : Block Diagram of the Fifth PWM Based DAC, Including Preceding DSP Stages.

2SCPWM with enhanced sampling was chosen for its good distortion performance and low output PRR, although this forced a zero interleaved noise shaper to be employed. For tonal intermodulation at a maximum of -100 dBFS, ten times oversampling can be used, yielding an output PRR of 220 kHz (triple tone test of 250 Hz, 4 kHz & 6.6 kHz @ -10 dBFS, worst IMD: -102 dBFS). For a worst case input tone (6.6 kHz @ -1 dBFS) this yielded distortion at -107 dBFS concentrated in the third harmonic, which was considered acceptable. Pseudo-natural pre-compensation was also checked under these conditions and was only 4.5 dB better, so this was not chosen because of the far higher complexity involved for such a small benefit.

With a maximum clock rate for commercially available CMOS logic at circa 100 MHz, eight bits can be counted from the noise shaper, fixing the fast counter rate at ≈113 MHz (127 MHz was chosen to permit an integer ratio between the fast clock and the interpolator bit-clock, allowing 32 high and 32 low guard band cycles). Optimisation of a noise shaper suitable for this was carried out and a seventh order design proved sufficient. Simulation of this noise shaper with the PWM parameters as above confirmed correct operation with minimal noise-noise or noise-carrier IMD as shown below :

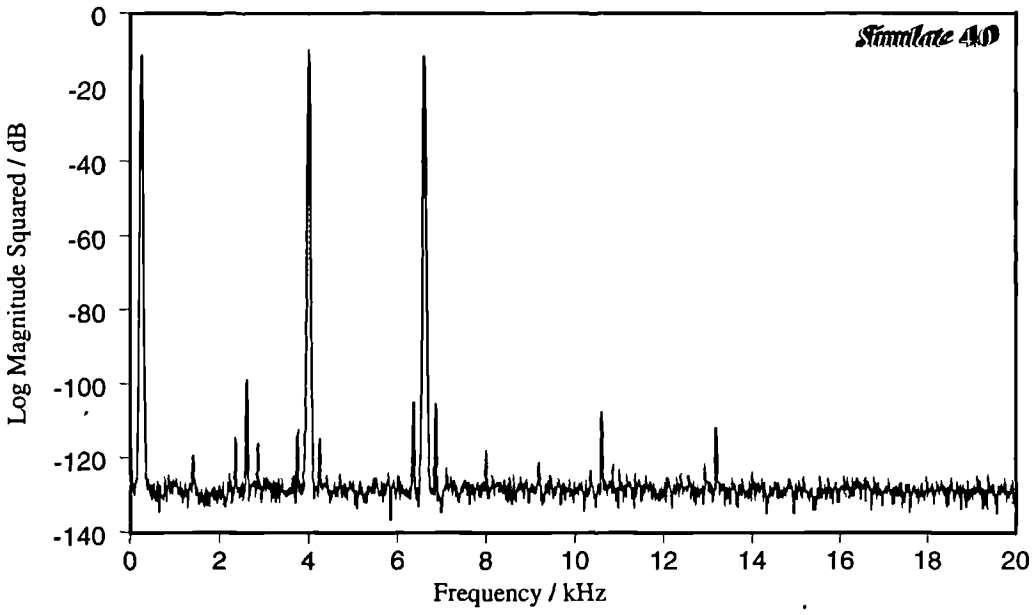


Figure 6.4.8.b : Simulated Performance of the Fifth PWM Based DAC (Triple Tone i/p - see text).

By using ASRC at the PWM DAC input, an eight times oversampling IC could be used with the remaining factor of 1.25x being achieved within the ASRC to provide ten times oversampling in total. This also ensures that any of the commercially available sample rates could be accommodated at the input (32, 36, 44.1 or 48 kHz) without exceeding the maximum interpolation ratio provided in the ASRC (2x) or relying on the ASRC filters to provide an anti-aliasing function as well. Furthermore, clock recovery at the input could be allowed since any jitter from this would be 'ironed out' by the ASRC and would not affect the fast counters within the PWM. Thus, a standard interface IC was used at the input (following the Sony-Phillips standard :'SPDIF').

Stereo circuitry for the above PWM was designed, with shared guard band circuits and clock division, and using the mixed asynchronous/synchronous techniques as in the third prototype. Serial communications and complementary signals were used wherever possible to reduce interconnection complexity and signal dependent power supply loading; double supply regulation was provided for the output re-synchroniser. Schematics for all these are included in appendix A7.

A power switch for use with this design could use full-bridge, class-D switching since both channels have complementary outputs. It was anticipated that a MOSFET power switch would be used, with resonant edge switching for low loss and minimal radio frequency emission, but the signal level circuitry has been designed to be general so that almost any type of class-D switch could be employed.

Spectral measurements from this circuitry are shown below after anti aliasing and notch filtering to remove the carrier, sidebands and the signal fundamental. These show the high dynamic range that even simple logic gates can support. Figure 6.4.8.c shows the signal band distortion arising from a worst case tone of 10 kHz, demonstrating both the low noise level in-band through the optimised noise shaping and the virtual absence of 2 nd. harmonic distortion through using 2SCPWM.

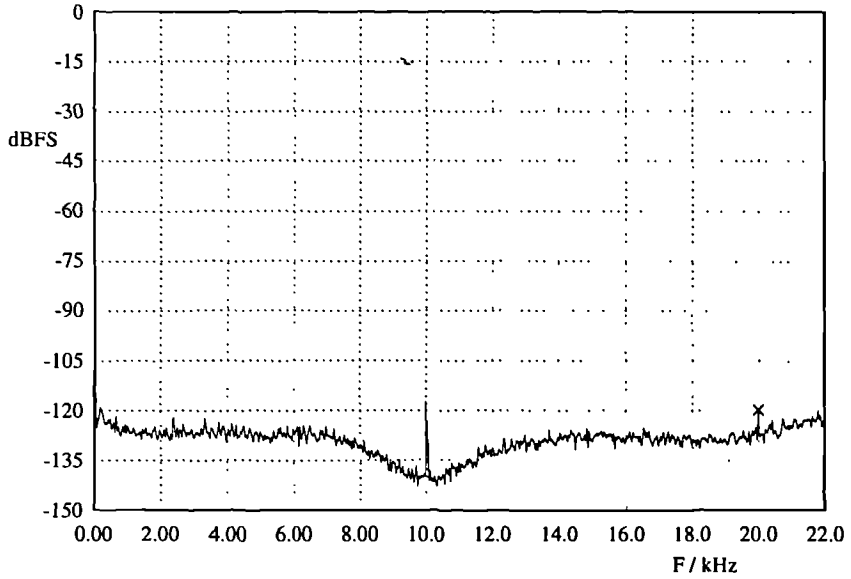


Figure 6.4.8.c : Measured Performance of the Fifth PWM Based DAC (Worst case tone - see text).

Below, the simulated and measured 3rd harmonic distortion level is plotted as a function of input frequency (with a -0.34 dBFS tone). In the hardware more distortion is present probably as a result of power supply variations, clamp diodes in the output of the logic gates, and anti-aliasing and notch filtering (as required to make a measurement with a low resolution A/D). Assembly language code to perform enhanced sampling and optimised noise shaping as used in these tests is included in appendix A9.

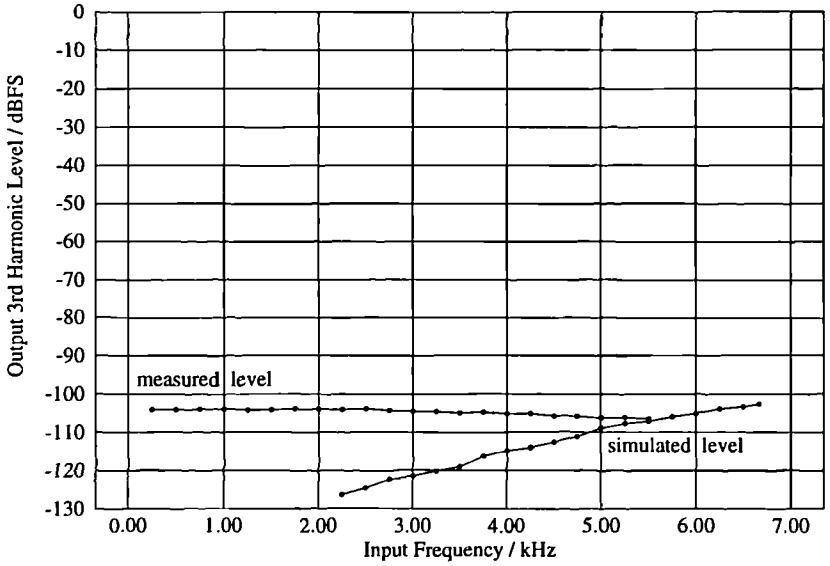


Figure 6.4.8.d : Summary of Performance of the Fifth PWM Based DAC.

6.5 : Preprocessing for High Speed DSP.

6.5.1 : Guard Band Circuits.

Guard bands were introduced into the third DPWM prototype to take advantage of using 288 fast clock periods per sample rate period. These were also carried through into the fourth prototype. Using 8 bit data allows control of 256 of the 288 periods providing 32 spare cycles. To maintain DC symmetry these should be split equally between the high and low portion of the pulse's allocated period. This implies sixteen cycles should be added to the high portion of the pulse and sixteen to the low portion.

To allow for proper symmetry in the pulse, the sixteen high cycles should be added to the centre of the pulse thus preventing a time shift of the pulse with its value (phase modulation). To avoid two low guard band circuits, the low cycles should be placed symmetrically around the timing marker. This presents a problem that the low guard band from one pulse is required to interfere with 8 cycles of the previous sample period. In order to allow pulse borrowing in a quantising noise shaper overload it was decided that by controlling only the eight low cycles at the beginning of the sample period (and leaving the 8 from the previous cycle, 'idle') this problem could be avoided. Since the data value is naturally limited, a pulse RESET should have already occurred leaving the 8 cycles in the previous cycle in the low state anyway. A timing diagram of these operations is shown below.

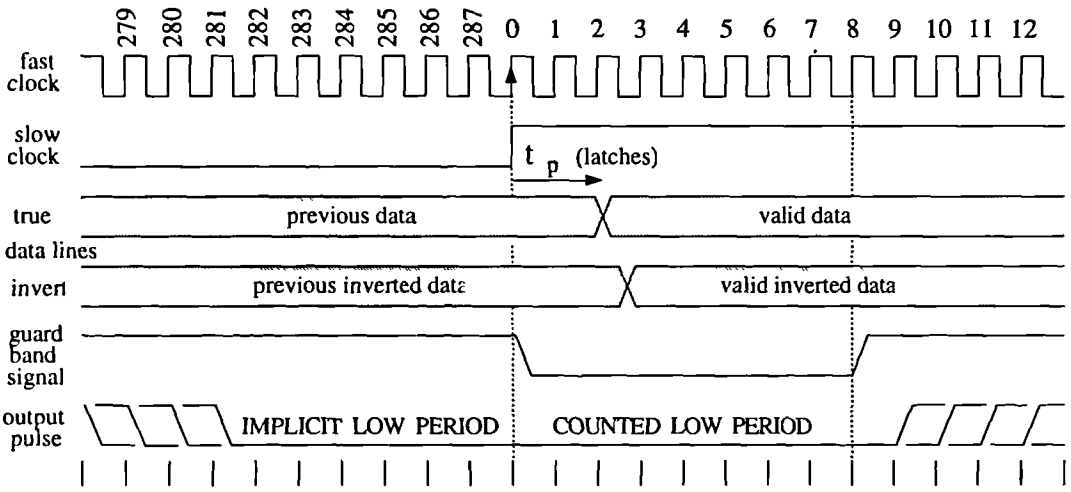


Figure 6.5.1.a : Low Guard Band Timing Control And Data Latching.

To insert 16 high cycles in the centre of the pulse presents difficulties requiring either larger wordlength counters and addition into the data, or high speed stopping and restarting of the counters which spoils the EOC detection circuit as used in the ASIC DPWM design. To avoid this, a 'floating' guard band was added which is inserted at the beginning of the pulse and moves with the pulse's leading edge as defined by the delay counter. This means that if loading of the counters is allowed during the guard band, data must not change from the beginning of the sample period (the low guard band and delay counter load time) to 8 cycles after the pulse centre (the end of the high guard band and the width counter load time). This prevents data from changing in the latches on the falling edge of the sample

rate clock (the sample rate period and pulse centre) at a time when the high guard band may still be allowing loading of the counters. The data latching edge must be moved to the start of the sample rate period as shown in figure 6.5.1.a, contrasting with previous designs where it had to be moved to the second half of the period. The timing of the high guard band is as shown below.

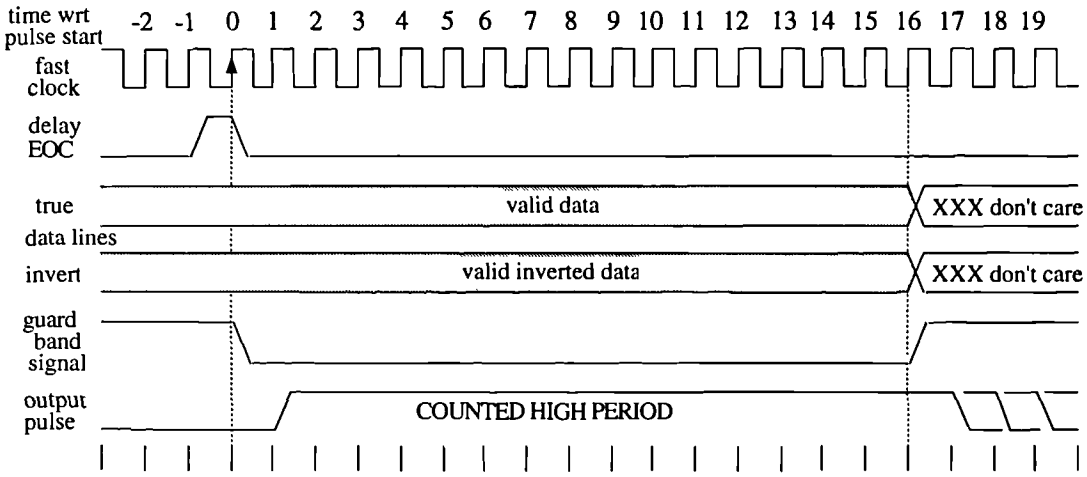


Figure 6.5.1.b : High Guard Band Timing Control.

As in the ASIC DPWM design, the delay counter EOC is delayed before actuating the output latch to accommodate the 'lost' clock edge used to release the width counter into counting. Should the data value be zero, the width counter EOC will be ready to 'RESET' the output latch as soon as the guard band is over, hence the entire output is delayed one fast clock cycle but the minimum pulse width is still 16 cycles. To ensure this operates properly, the guard band circuit has to be triggered by the delay counter EOC, not the output latch 'SET'.

As can be seen in the two timing diagrams (figure 6.5.1.a & b) the relevant counter must be able to commence counting as soon as the guard band time is over. Control of the clock requires gating at 100 MHz which introduces delay making synchronisation of the various parts of the circuit very difficult (as experienced in the second prototype). To avoid this, the LSB flip-flop of the counter which the guard band circuit controls is held from changing by SET-ing its J and K inputs. Thus the clock can be left running with the entire counter in a 'HOLD' state. This does not affect its asynchronous inputs which can be used to force the state of the output and effect loading of the data LSB.

Since the EOC detection circuit must be given direct access to the next latch's synchronous inputs, spurious EOC signals generated during the load time (when the data has not yet settled), have to be covered by further logic. This was achieved by asynchronously forcing the state of the output latch during the guard band time. Knowing the state of the guard band being controlled allows the low guard band circuit to force a 'PRECLEAR' on the output latch and the high guard band circuit to force a 'PRESET'. The preset and preclear signals for the 74AC109 have a negative recovery time (w.r.t. the clock) so these signals can be removed from the output latch slowly without any danger of upsetting the EOC that they are covering. Since the delay counter EOC is delayed before reaching the output latch, this covering logic should be prevented from overtaking the synchronous signal by not starting until the load time which is kept shorter than the guard band duration.

To maintain fully synchronous operation a control latch was used to switch operation of the guard band circuit's internal counting and waiting and the delay (or width) counter's internal counting and waiting (to 'HOLD' during loading). By connecting the controls to these two sets of counters to the same control latch, complementary count and wait times could be assured in both as shown below.

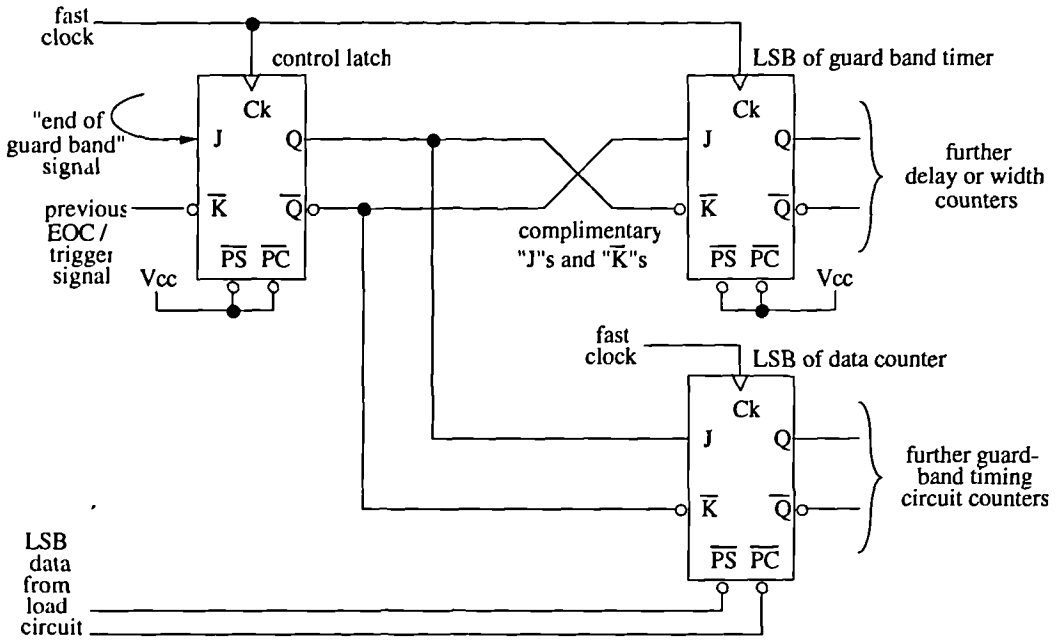


Figure 6.5.1.c : Control Latch Configuration For The Guard Band Counters And The Data Counters.

As the EOC detection circuit must be allowed time to settle, the load time has to be finished at least the settling duration before the end of the guard band. Furthermore, in the high guard band time the load time must not start until after the output latch has been synchronously SET (by the delayed EOC from the delay counter). By using inverse logic in the last flip-flop of the guard time counter, this stage will be active in the middle half of the guard time rather than the second half thus enabling both of the above constraints on the load time to be met by this signal. This causes a race hazard in the guard time EOC detection, but because of covering logic in its LSB stages, this does not propagate through to its output. A partial schematic of one guard band timing counter is shown below:

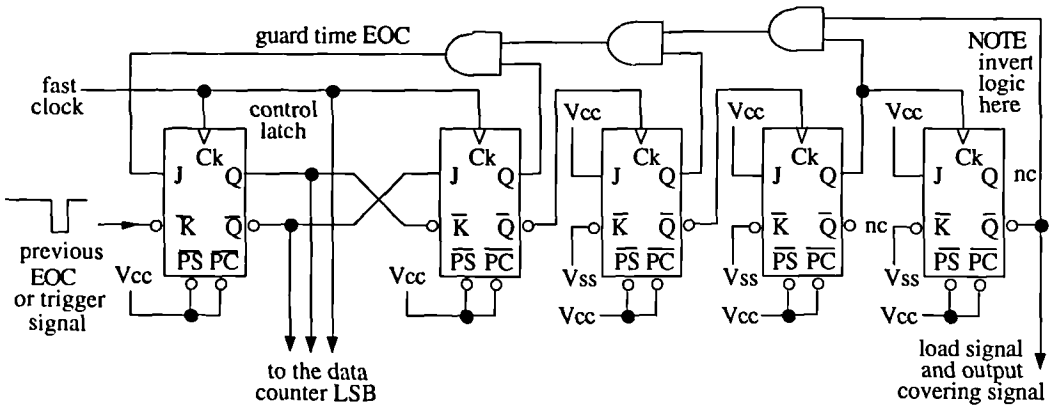


Figure 6.5.1.d : High Guard Band Timing Counter.

It is useful to note that the points in this circuit where the fan-out of the counting elements is increased are the penultimate and last flip-flops which are also the slowest clocked devices. Small fan-out in the LSB of such counters and EOC signals greatly assists making the logic count at 100 MHz and beyond (the advertised maximum operating speed for this series is 60 MHz).

Problems can occur when a zero value is entered into the data counter (-128 in 2's complement notation) since the EOC logic will output a RESET to the output latch which will be responded to as soon as the covering logic signal is removed (4 cycles too early). This can be covered by further logic as shown below although this has not yet been considered necessary because to maintain symmetry in the signal, data values of +127 and -127 are the data limits. This additional logic loads the guard time EOC, making the circuit not operate quite as fast, so this was not included in the design.

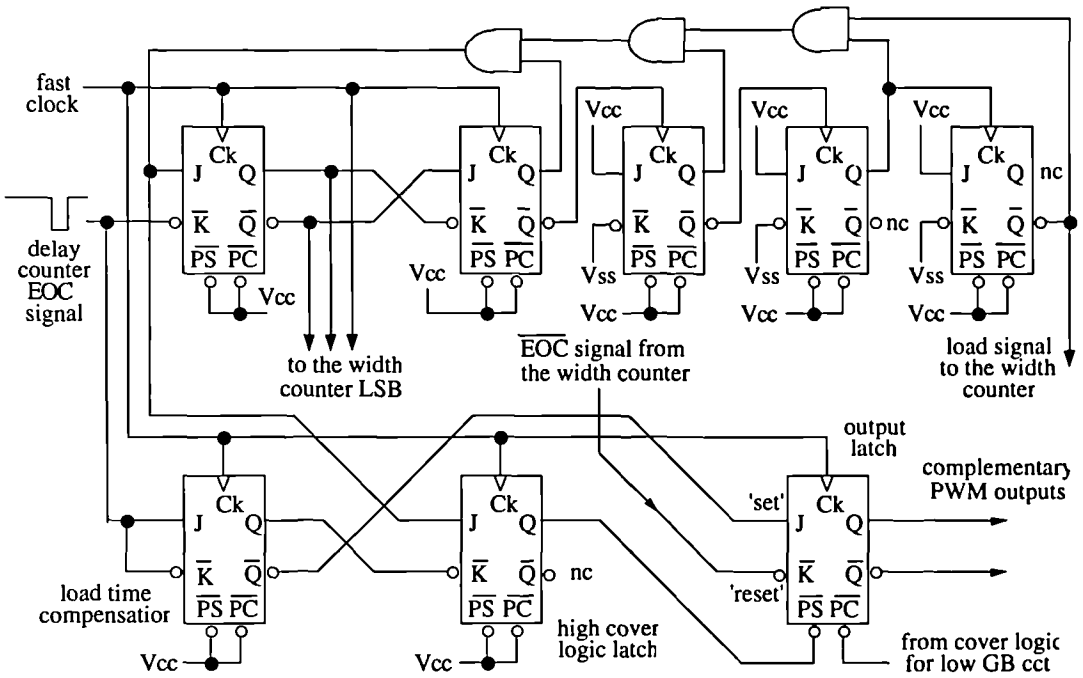


Figure 6.5.1.e : Covering Logic For Inclusion Of The -128 Data Value

The guard band circuits were simulated using ORCAD logic simulation software. Full schematics of the design as built are provided in appendix A7.

6.5.2 : TEPWM, LEPWM, LAPWM & AOAPWM Load Circuits.

Simple data bus handling can be used for all the one-sample-per-pulse modulation types with all of the modulator prototypes discussed in section 6.4 .

In the case of the trailing edged modulation, data for the delay must be held constant. Any value can be used for this which allows full modulation of the the remaining time in the sample rate period. Taking DPWM III as an example, this would imply loading a value greater than 247 (255-8) into the delay counter on every cycle to avoid the pulse being curtailed by the next guard band period (the spare ports are zeroed by resistor pull downs, if left unconnected). Then the width counter can be loaded with data as normal.

Leading edged modulation can be achieved by loading the width counter with the same as that loaded into the delay counter forcing the pulse finish point to be held constant (since the counters are complementary: one up one down).

Lagging asymmetric PWM was wired into the third and fourth prototypes and true symmetric PWM was provided in both the first and second prototypes. Signal harmonic performance for these two modulation types is so similar that the additional clock speed to create true symmetric was though not worthwhile in the later designs, although small noise floor differences do exist. Both types involve programming the delay time to be half the low time in the sample period, so that the pulse is centred. This can be done by simply dividing the data by two to define the delay since the delay counter has already been configured to be an up counter (a bus priority shift).

The LSB is discarded in the LAPWM case which leads to both odd and even valued pulses starting after the same delay, shifting the pulse by an additional half cycle in the odd case. If the LSB is used to advance the clock phase by 180° (inverted), the pulse can be re-centred yielding the true symmetric modulation type. This was attempted in the first two prototypes but was found to make the counters more difficult to control at high speeds, and precise control of the clock phase also proved difficult.

Retaining the LSB information and alternately adding it to the leading and trailing edge whenever it occurs yields alternating asymmetric PWM. This requires additional logic circuitry to 'remember' which edge to put the asymmetry on depending on which edge was used last time an odd value cropped up. By adding '1' to the delay counter data, the pulse can be brought earlier into the sample period to create leading asymmetry; otherwise, truncation of the width data in producing the delay causes lagging asymmetry in the pulse. A latch toggled whenever odd data is admitted, can be used to add into the delay data through the carry in of an adder such that the LSB is alternately counted twice. Thus after division by two, the LSB is alternately rounded up or down instead of being always truncated off (and subsequently ignored). Since generous guard time is available within the third and fourth prototypes, this addition into the data stream can be performed fast enough to be completed before loading commences, completing the AOAPWM control. The difference this makes to the SNR is dramatic, since the asymmetry is at the 8 th. quantised bit (for an 8 bit modulator) which is very significant compared to the LSB of the input data (16 th. bit for a CD). Shown overleaf are the spectral performance of TEPWM, LAPWM and AOAPWM with a 1 kHz tone using the same interpolator, noise shaper and oversampling ratio. The degradation of the noise floor in the LAPWM case is clearly visible; in the AOAPWM case, the noise floor can be seen to have been restored (back to that possible in this system as shown by the TEPWM performance)

Detailed schematics of the circuitry required for this type of modulation are included in appendix A7.

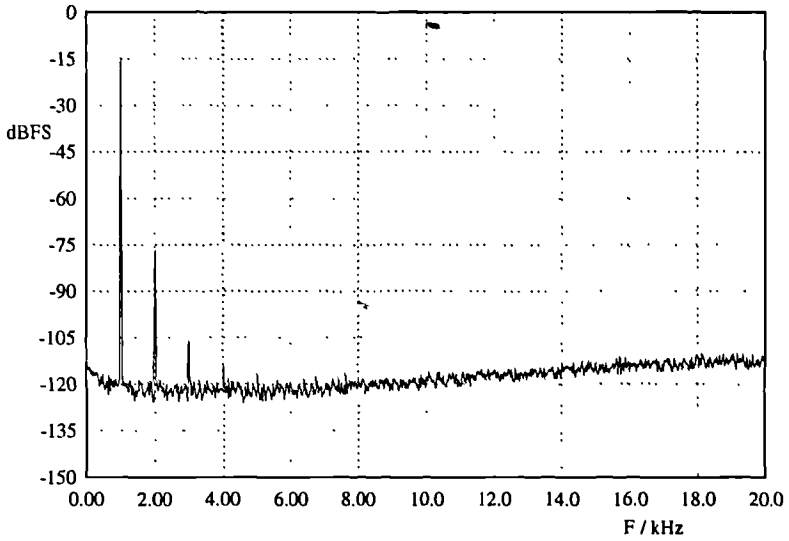


Figure 6.5.2.a : Spectral Performance Of TEPWM Showing A Typical 16 Bit Noise Floor.

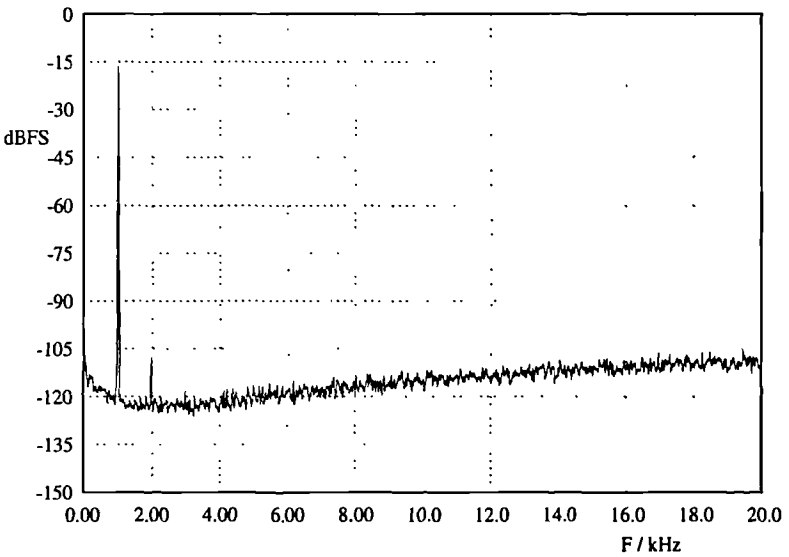


Figure 6.5.2.b : Spectral Performance Of LAPWM Showing Noise Floor Degradation.

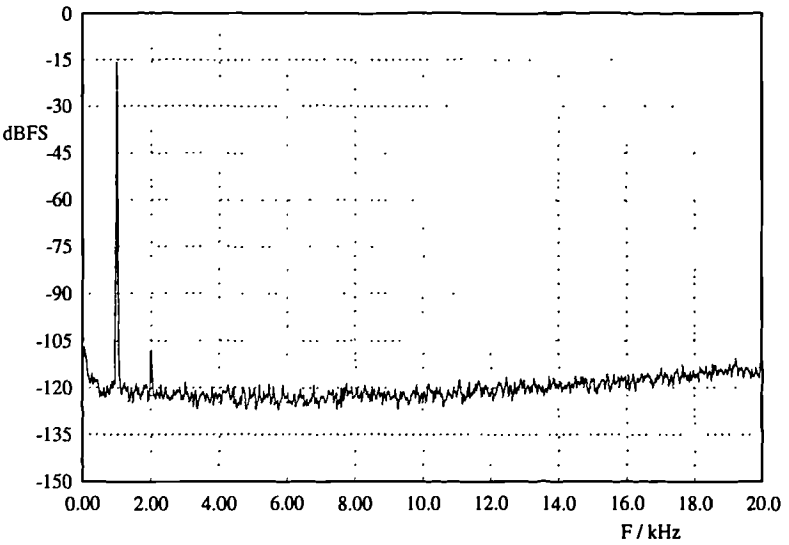


Figure 6.5.2.c : Spectral Performance Of AOAPWM Showing Restoration Of The Noise Floor.

6.5.3 : 2SCPWM Load Circuits & Clock Control.

When two samples per pulse are used with the same fast clock rate, either the pulse repetition rate has to be reduced or the wordlength of the samples included in one sample period have to be halved to fit them both into the time available.

Reducing the wordlength can only be achieved if the noise shaper can be re-configured to output fewer bits and handle zero interleaved NTFs (see section 4.5.3). This increases the noise power (fewer bits) and introduces the potential for noise sidebands in the audio band. Reducing the pulse repetition rate also increases the severity of the sidebands, so special noise shaping is required in both cases [HIO93].

Provided the data has been appropriately prepared, consecutive sample pairs must be added together to form a new width value, and the first of the two samples must be used to define the delay. Since the delay value is the direct complement of the first half data no division is encountered (or its associated truncation). These operations can easily be included within the special noise shaper required for the preparation of the data. For this reason, they have not been included in hardware. The opportunity to divide the clock was provided in the fourth prototype so that 2SCPWM could be implemented with data preparation being carried out on a separate DSP PCB. Schematics for this are included in appendix A7 along within those for the fourth prototype.

6.5.4 : DSP Based Pre-compensation & Floating Point Noise Shaping.

Pre-compensation and floating point noise shaping both require the use of DSP techniques which are complicated to implement in discrete devices. DSP algorithms running on a general purpose DSP IC were developed to accommodate WAPWM, 2SCPWM, ESPWM, PNPWM and low power gain noise shaping (to reduce sideband problems).

Two target systems were used for these algorithms : a DSP96002 processor running at 33 MHz and a DSP56004 running at 40 MHz. Many algorithms could be implemented and that for WAPWM will be used as an example of the kind of operation that can be achieved with such a processor. Details of the ESPWM and noise shaping assembly language routines can be found in appendix A9.

For WAPWM, the estimated intersection of a sawtooth with the signal is calculated from the gradient between two adjacent samples and the first sample value (see figure 5.3.2.b). Scaling and division are also required although the scaling can be precalculated and held in register, and the division can be applied by repeated nested multiplies to approximate the true dividend (see equation 5.3.2.e).

An algorithm for this was written and the data produced by the DSP card was fed in real time to the fourth order cascade noise shaper and subsequently to the third DPWM prototype configured as a TEPWM. The spectral performance of uncompensated TEPWM and the above system are shown overleaf for a 1 kHz tone, at -6 dBFS. These analogue tests of the low pass filtered signal level output have been notch filtered to enable high resolution measurement of the harmonic distortion level.

The second harmonic level of the standard trailing edge modulator is as simulated (at -62 dBFS) which agrees closely with theory. The second harmonic in the pre-compensated signal is some

25 dB better (-87 dBFS) but not as good as theory would suggest. Various tests have been applied to find out what the cause of this discrepancy is. Potential culprits are, 1) incorrect scaling and guard band allowance in the algorithm, 2) interpolator spectral replicates, and 3) power supply variations. Re-reading of the data back into the simulator for analysis suggest that (3) is the problem although higher dynamic range measurements have been taken with AOAPWM (eg. figure 6.4.6.i).

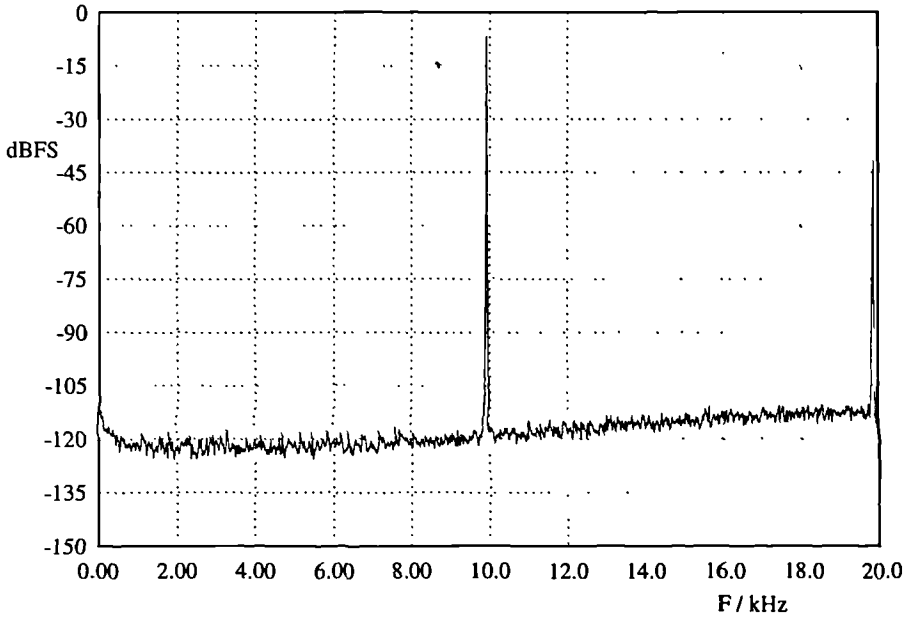


Figure 6.5.4.a : Standard TEPWM Spectral Performance, -6 dBFS, 10 kHz Tone.

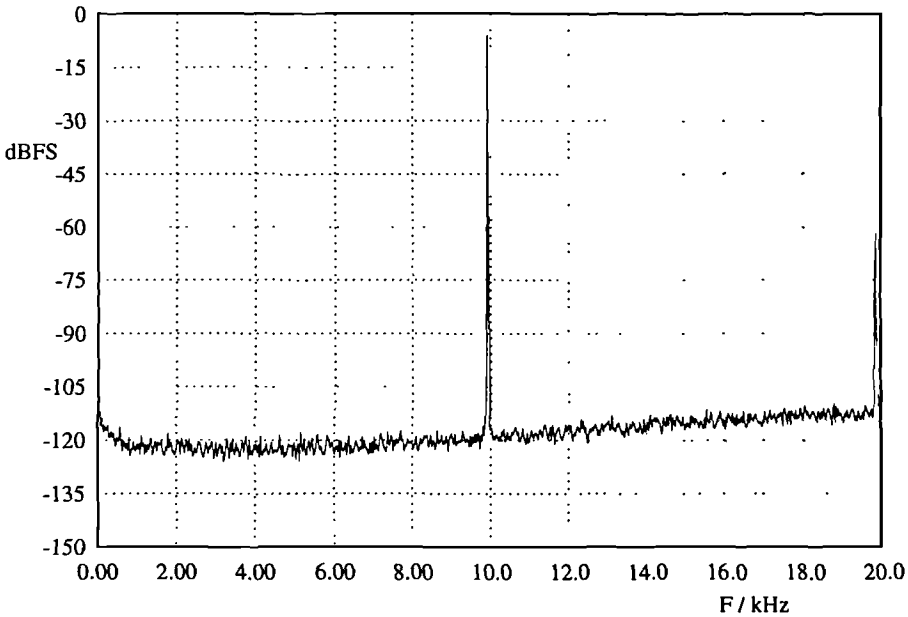


Figure 6.5.4.b : WAPWM Spectral Performance, -6 dBFS, 10 kHz Tone.

6.6 : Summary.

Circuits have been designed and built to perform sinusoidal noise shaping (first to fourth order) and all known digital pulse width modulation variants (DPWMs). These circuits have been interfaced to stored, precalculated, and real time sources such as a CD player and standard digital signal processors (TMS32020, DSP96002 & DSP56004). Output has been measured and listened-to through low pass filtering, at both signal level and at up to 2 W power level. Using a signal processor, pre-compensation for the DPWM non-linearity has been applied and shown to be effective. Floating point noise shaping can be applied within the same processor, instead of a sinusoidal noise shaper circuit.

First or second order sinusoidal noise shapers were constructed in application specific integrated circuits (ASICs), which were subsequently cascaded for higher order systems. Input data with sixteen bit wordlength can be shaped to yield output wordlengths of between one and sixteen bits at sample rates up to c.20 MHz. Programmed array logic (PAL) was used to configure order and output wordlength in these devices through a serial port. Digital measurement showed these devices performed very similarly to both theory and simulation. Maximal-length-sequence, single-bit, additive dither was used to decorrelate idle channel tones and small signal distortion.

Five DPWM prototypes were constructed, each developed from its predecessor to satisfy more advanced requirements. Special counter designs were investigated to allow high speed counting in commercially available logic. Advanced CMOS discrete logic was coerced into operation at three times its maximum manufacturer's family speed by using mixed synchronous-asynchronous design. Operation at a clock speed of 180 MHz was achieved, although lower speeds were preferred for interface with other circuits. Operation using a 101.6064 MHz quartz based clock was extensively researched with 8 bit modulation and 8x oversampling (Fs=352.8 KHz). Results show that a PWM DAC to complement CD quality can be achieved (16 bit resolution, 20 KHz bandwidth). For example, THD for a 1 KHz tone at -15 dBFS was measured as 0.003% and noise floors of c.-120 dBFS were achieved using alternating odd asymmetric modulation and fourth order sinusoidal noise shaping.

Clock phase control was rejected as a technique for increasing resolution, preferring simple clocking for its lower jitter performance. Guard bands were found to be a necessity for good modulator performance, and these enabled simpler loading circuits to be employed. To reduce pulse phase modulation, re-synchronisation of the output is required and to reduce pulse amplitude modulation, complementary outputs were found useful. Independent output power supplies were also employed to reduce pulse amplitude modulation, primarily by suppressing supply borne noise. Mixed switched and linear regulation of the power supplies was used to remove line frequency interference, although other interference indicates that screening is required for the high accuracy parts of the circuits.

Serial and parallel interface techniques were used to connect into a modified CD player source, driven by a sub-division of the fast PWM counter clock. First-in, first-out memory was used to allow asynchronous operation between the source and the PWM DAC. 8x oversampling was imitated by digital interpolation but lower pulse repetition rates are recommended for future implementations (5x).

Sample rate clock multiplication to produce the fast DPWM counter clock was used in the first prototype; this technique was replaced in experimental designs because of its high jitter. Instead, a local oscillator and asynchronous sample rate conversion is recommended for any commercial design.

Chapter 7 : Output Switch and Filter Design

7.1 : Switch & Filter Requirements.

7.1.1 : Overview

The use of switching output stages after a PWM DAC can theoretically provide high efficiency conversion (as a result of a class D output) as well as high accuracy conversion (if a crystal based oscillator is used). Until recently, the difficulties of high accuracy conversion have been considered most significant since low accuracy conversion can easily be achieved by conventional techniques (eg. successive approximation or current summing DACs). High accuracy conversion can be obtained with PWM by using natural sampling but since this has been thought to be only available in analogue PWM systems [SAN83] it has not been used in except in its analogue form. Analogue pulse width modulators have noise limited performance which has restricts their use to applications where efficiency is of primary concern (eg. public address). Hence digital PWM is not previously been used for power D/A converters.

By using pre-compensated modulation types (eg. WAPWM, ESPWM, PNPWM) the distortion problems of DPWM systems can be reduced to competitive levels or beyond when compared to analogue amplifiers (ie. class A, B or A/B). With the use of a high efficiency switch, producing a high power signal should be possible, theoretically with the same distortion as at the signal level stage provided the power switch itself does not distort the signal significantly.

The design objectives of power level D/A converters are considerably less stringent than for signal level D/A converters since the power stages of analogue amplifiers are not totally linear and the transducers used for sound reproduction from the amplified electrical signal (usually loudspeakers) are seldom distortion free either. This may allow the use of sub-optimal power switching schemes or poor modulation types and hence a DSP cost reduction, or conversely, allow an improvement in the best possible power D/A conversion performance. The efficiency improvement that class D output stages offer enables both high power amplifiers to be constructed with reduced heating of output devices and lower power amplifiers to be constructed which can conserve their battery supplies; thus both large heatsinks can be avoided and weight and power supply requirements can be reduced.

Many factors have to be considered in the design of the output switching stage which can be split into two groups : those created by the modulation choices (ie. AD or BD PWM, one edged or two edged modulation, modulation depth, counter speed and pulse repetition rate) and those created by the output requirements (power level, variable or fixed amplitude output pulses, load impedance, EMI suppression). The implications of these factors will be discussed in the next section, the circuit performance shortfalls will be discussed in section 7.2, the switch and driver configurations will be discussed in section 7.3, and the output filter design and configurations will be discussed in section 7.4.

7.1.2 : Choice of Switching Frequency

Choosing a pulse repetition rate (PRR or ω_c) for output switch design is a balance of several factors some of which come from the modulator, its compensation and noise shaping, others of which come from the circuit topology and devices used in the output stages. In a simple PWM-based power DAC the choice of PRR is limited since it has to be the same as the sample rate of the noise shaper (or 1/2 that for two samples per pulse). For this case, only reducing the modulation depth (scaling of the input) and increasing the pulse amplitude (scaling of the output) can be used to control the amount of time available for switching transitions and the level of distortion in the output. Such a technique reduces the output efficiency significantly.

Unless asynchronous sample rate conversion (ASRC) is employed, the sample rate in the DSP stages (eg. the noise shaper) has to be a rational factor higher than the input sample rate, preferably an integer factor, and ideally an integer which is even. Even-valued integers permit the initial (and toughest) interpolation stage : up-sampling by a factor of two, to do only the minimum required filtering, and many devices have been developed for this (eg. 4x and 8x oversampling are common). ASRC devices can reduce this restraint, since fractional sample rate change can be performed (between 0.5 and 2x [ADI93]) as well as operation with asynchronous signal sample rates and PRR which allows the use of a separate, low jitter, counter clock. A combination of ASRC and synchronous interpolation would thus be recommended.

For the remainder of this chapter, CD sources (16 bit, F_s=44.1 kHz) will be assumed; sources with slightly different sample rates can be interfaced by using ASRC so the output filter requirements for various PRRs shown below are only for the common (power of two higher) sample rates. Such output filters are similar in nature to recovery filters as used in conventional DACs where oversampling is used to enable simpler filter characteristics (as discussed in section 3.4.1).

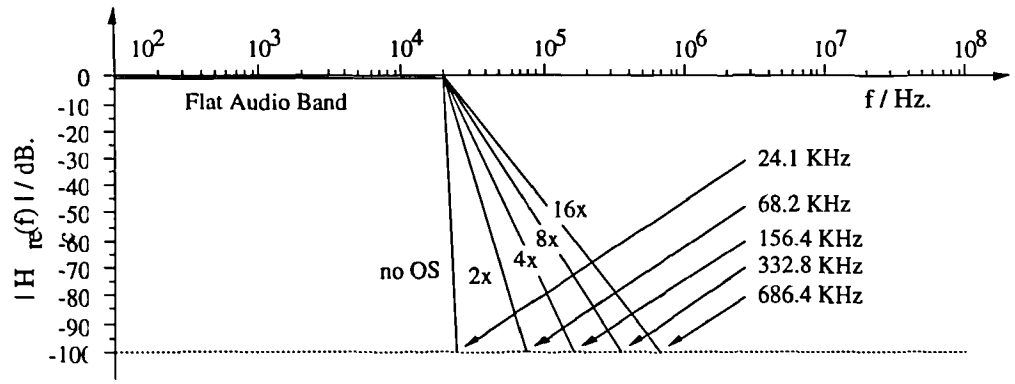


Figure 7.1.2.a ; Recovery Filter Requirements for DACs With and Without Oversampling

Recovery filters become easier to implement as the sampling frequency increases since the filter order reduces dramatically as the transition from passband to stopband widens. The recovery filter order reduces diminishingly with oversampling, so while at least four times oversampling is worthwhile, the increased interpolator and pre-compensator complexity in using oversampling ratios higher than 16x would suggest that such ratios are not used.

From theoretical DPWM studies (see chapter two), the sidebands of the first carrier are expected to appear in the baseband above the quantisation noise floor for CD quality signals if ω_v/ω_c is less than 1/10.25 (worst case, @ 20 kHz, PNPWM with OS=5x). While this is a stringent test and for competitive power DACs, 4x oversampling might be considered acceptable, it is suggested [HIO91b] that for high resolution power DACs employing WAPWM or PNPWM, the minimum OS ratio is 6x (the next larger, even, interpolation factor), leading to a minimum PRR of 264600 Hz. If 2SCPWM or DSPWM are used (with or without further pre-compensation) the PRR can be kept at 264.6 kHz, but the OS ratio would double to 12x as a result of needing twice the data rate per pulse.

Although the modulator will become more linear as the signal to carrier frequency ratio increases, non-linearity in the output switch quickly overtakes this, so pulse repetition rates (PRR) greater than 1 MHz are not useful (c.20x OS). Beyond this point, class D output stages struggle to handle suitable power levels (tens to hundreds of watts) and become ridden with distortion from pulse interaction making them unsuitable for audio use [ATT83].

A more stringent constraint than the efficiency limit may be that of signal to noise ratio (SNR) which becomes degraded by noise sidebands in open loop DPWMs using noise shapers whose output contains high noise power before modulation. In highly oversampled systems (eg. 20x), the maximum counter clock rate can force using fewer bits in the modulator, which despite lower gain in the noise shaping can lead to greater sideband problems. For example, a 100 MHz clock limit would imply using a six bit modulator (for AOAPWM, eight for 2SCPWM) which in simulation, led to an audio band SNR of only 78 dB with the best minimum power gain optimised NTFs (section 5.1.1).

Conversely, at low oversampling factors, the near frequencies of the shaped noise have a higher power which induces higher levels of signal-noise intermodulation, also degrading the SNR. Experimenting with NTF shapes for evading noise sidebands in the audio band suggest only small improvements can be made unless higher counting rates are permitted in the DPWM itself (ie. by using higher cost implementations such as emitter coupled logic, ASIC or MMIC)

To summarise, ratios of between 6x and 20x will be considered feasible with higher efficiency available at the lower rates, and higher accuracy available at higher rates.

7.1.3 : Pulse Edge Timing Accuracy Requirements

System design and simulation to this point have assumed output pulses with square edges placed exactly at points in time defined by an appropriate clock; in reality this is not the case. Error is introduced by limited slew rate during a transition, delay between the intended switching instant and the time at which the output responds, the edges having non-ideal shape such as mid-height plateaus or overshoot, and the edge time wandering with respect to the 'ideal' timing markers. These errors can be separated into two parts : those that are static and introduce an offset into the output, and those that vary (either randomly or as a function of the signal) which introduce noise and distortion.

The static effects are those such as the limited $\partial v/\partial t$ of the output which is commonly band limited by the parasitic inductances of device packaging and the stray capacitance of output stages' layout. While these do modify the pulse edges, changing the pulse area, and hence the energy passed to the load, the baseband consequence is commonly an offset and their high frequency consequences can

usually be ignored (reduced harmonics of the carrier and their sidebands). Theoretically, additional offset circuits can be used to restore the pulse area although these must be allowed for if pre-compensation such as WAPWM or PNPWM is used (these are non-LTI, hence offsets must be allowed for). In some cases, reduced harmonics of the carrier and their sidebands could be said to be a good 'fault' since the reduction of high frequency components reduces radiated EMI and may be a design feature to include in future designs in the light of tightening controls for electromagnetic compliance (EMC).

The varying effects are predominantly the uncertainty of the edge timing and the increased difference between the output switch turn-on and turn-off delay as a function of load current. The uncertain edge timing is of serious concern (and will be discussed in depth shortly) since this directly alters the pulse duration and sets a limit on the performance that can be achieved with any DAC using pulsed output (eg. MASH, ΣΔ, PEM). PWM is considered to be one of the most tolerant of possible systems because it uses a minimum of edges per second, but inappropriate edge timing can restrict output signal quality from the PWM based DAC if not properly avoided by design. The turn-on/turn-off delay difference which is a function of load current, can be reduced by pre-lengthening or pre-shortening the pulse as a function of the input signal if the load impedance is constant. This task is most suited to a DSP based implementation, and would be a good subject for future work; this will not be discussed further here. Other non-static non-idealities such as the output overshoot can be made largely independent of the signal by appropriate use of guard bands and will not be discussed further.

Despite the pulse edge resolution being reduced by noise shaping, the pulse edge timing accuracy has to be high for the full input resolution to be achieved. This implies that each pulse area has to be defined at least as accurately as the equivalent PAM pulse would be, if the energy passed to the load is to be modulated as intended. Variation between the intended switching time and the real switching time directly changes the pulse area (and hence the energy passed to the load). Such a change arises from two sources : error in the clock timing, and time offsets added by subsequent circuitry. These will be referred to collectively as pulse edge jitter. In the remainder of this section the size of jitter which can be tolerated will be estimated, a crystal clock source will be measured, and the effects on a PWM output from the injection of gross jitter will be measured and simulated, to demonstrate the approximate system requirements.

Noise shaping requires a PRR higher than the Nyquist sampling frequency for a given bandwidth and if high enough this can avoid sidebands in the signal band. Unfortunately, this increases the number of edges per second and hence introduces more opportunity for error. To assess the jitter that can be tolerated, the oversampling ratio must be known (and hence the number of edges); then a comparison between the jitter noise and the inherent quantisation noise of the input can be made.

From Equation 4.3.3.e but substituting amplitude 'A' in place of 'X' the inherent quantisation noise power, σ_{qn}^2, assuming a sinusoidal, 'b' bit wordlength digital signal can be calculated :

$$\sigma_{qn}^2 = \frac{A^2}{12.2^{2(b-1)}} \qquad \text{Equation 7.1.3.a}$$

This result will be used later for comparison with the error introduced by jitter, assuming interpolated data (where all the above quantisation noise power resides in the signal band). Although more edges are

used in an 'L' times interpolated pulsed output, the bandwidth over which this error is spread is also increased, so if the error can be assumed white the pulse accuracy need not be improved.

In contrast, for an 'L' times oversampled system the pulse period decreases by a factor of 'L', so the edge accuracy has to be improved to keep the jitter within 1/L of that expected for the signal before oversampling. This is equivalent to the additional 3 dB improvement in the quantisation noise power per oversampling doubling as available in $\Sigma\Delta$ ADCs. For example, if the signal is oversampled by a factor of two, the period decreases by a factor of two, and so the pulse edges have to be measured to twice the accuracy (ie. the pulse energy has to be controlled to twice the accuracy) so the output power is quantised twice as finely (3 dB quantisation noise power decrease).

Accurate assessment of the edge timing requirements is difficult since this can only be completed with a full knowledge of the modulating signal, the noise distribution and the modulation type. Two techniques can be used, the first based on simulation where the pulses are clocked by an oscillator advanced or retarded by an offset signal, the second based on a full PWM analysis as used for the results in chapter 2 (see appendix A.8) and [ROW65]. Both techniques are complicated, involving assumptions such as the noise distribution shape, which can introduce large inaccuracy.

Approximate assessment of the edge timing requirements can be made by simply relating the edge resolution to the amplitude resolution. For this technique to be used, the jitter must be assumed small since for an amplitude change 'Δv' in the instantaneous amplitude of the modulating signal, 'v', the output power change is :

$$\Delta P_{out} \ \alpha \ (v+\Delta v)^2 - v^2 = 2.v.\Delta v + \Delta v^2 \qquad \text{Equation 7.1.3.b}$$

whereas for a width change 'Δt' in the pulse associated with the above sample, the output change is:

$$\Delta P_{out} \ \alpha \ (\tau+\Delta t).A^2 - \tau.A^2 = \Delta t.A^2 \qquad \text{Equation 7.1.3.c}$$

By assuming 'Δv' to be far smaller than '2.v', 'Δv^2 ' in equation 7.1.3.b can be ignored, allowing each output power change to be proportional to its associated change ('Δt' or 'Δv'), and hence implying :

$$\Delta v \ \alpha \ \Delta t \qquad \text{Equation 7.1.3.d}$$

Later in this section, typical values for the ratio 'Δv : v' will be seen to be less than 10^{-5}, so the above assumption will now be used in an assessment of the permissible edge jitter for a CD quality signal.

Considering a directly modulated 44.1 kHz sampled 16 bit resolved signal, a pulse edge accuracy which allows distinction between 2^{16} levels in each sampling period (22.7 μs) is required. Splitting the pulse period into 2^{16} parts implies edge separation increments of 346 ps, suggesting that an edge increment accuracy of ± 173 ps is the limit of edge time jitter that can be tolerated without introducing intersymbol interference. If the jitter can also be assumed to be Gaussian distributed about the intended edge time (this is a poor approximation) the RMS jitter can also be calculated as below.

Two edges are used to define each pulse, and these can move independently or in a correlated way. Although approximate, the effects of errors in the pulse edges are simpler to calculate if they can be considered orthogonal, relating them to amplitude and phase errors in the eventual (low pass filtered) analogue output.

Positively correlated movement of both edges can be viewed as phase distortion since when both edges move in sympathy the pulse energy is passed to the load too early or too late. If negatively

correlated, the edge movement can be viewed as error in the amount of energy passed to the load which after filtering will appear to be amplitude distortion; it should be stressed that this is through modification of the pulse area, not modification of its amplitude. General movement of both edges will introduce both phase errors and amplitude errors.

For reasonably good modulators the error from the intended switching time can be assumed to be smaller than the period of the counter clock, thus the jitter in each edge position can be treated as a separate random variable. For analysis, the jitter will be defined as the deviation from the ideal edge time in units of clock periods of a clock at the product of the sample rate and the resolution. For example a nine bit modulator with a PRR of 200 kHz would have a clock period of $1/(2.0\text{x}10^5.2^9) = 9.766$ ns. For the first edge this jitter will be called 'X_1' and assumed to be Normally distributed, with zero mean and variance σ_1^2. For simplicity, the error in the second edge will be assumed to be statistically independent of the first, defined as 'X_2' and distributed similarly, $N\{0,\sigma_2^2\}$. Strictly, this independence assumption cannot be made unless jitter has no frequency content with a period larger than the minimum time between edges and the edges are separated by several times the coherence of the master oscillator; this is the main source of error in this approximate calculation.

Using these definitions, the modifications to pulses in terms of phase and energy can be expressed as shown below :

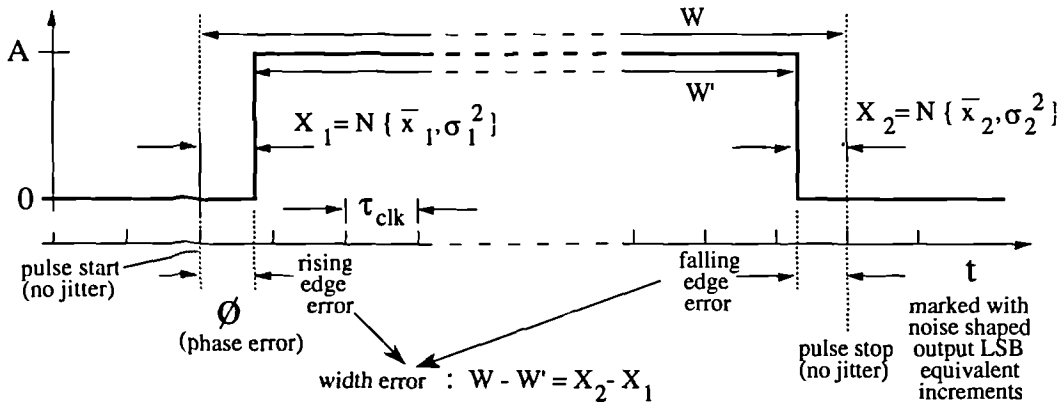


Figure 7.1.3.a : Pulse Jitter Definitions.

The phase error can be taken from the first edge and is X_1. For pulses of amplitude 'A', the amplitude error arises from the difference in the errors associated with each edge and is $A.(X_1-X_2)$.

Since the expected values of X_1 and X_2 are the same, the expected width error $(=X_1-X_2)$ is zero and so the expected pulse width is as intended, 'W'. Similarly, since the expectation of X_1 is zero, the expected phase error is zero.

The variance of the phase error measured in counter clock periods without noise shaping is simply that of the rising edge, ie. σ_1^2 and the variance of the pulse width error is the variance of the difference of distributions, N_1-N_2, which is $\sigma_1^2 + \sigma_2^2$. From this the RMS uncorrelated error voltage can be calculated:

$$V_{UEE} = A.\sqrt{(\sigma_1^2 + \sigma_2^2).\tau_{clk}/\tau_{PRR}} \qquad \text{Equation 7.1.3.e}$$

(where 'τ_{clk}' and 'τ_{PRR}' are the periods of the counter clock and the pulse repetition rate).

For a PRR of $L.f_n$ (where 'f_n' is the Nyquist sampling rate), the noise power contribution from uncorrelated edge error (UEE) is †:

$$N_{UEE} = L^2.f_n^2.A^2.(\sigma_1^2 + \sigma_2^2)/f_{clk}^2 \qquad \text{Equation 7.1.3.f}$$

As a result of the pulse moving by a phase error, $\emptyset = N\{0,\sigma_1^2\}$, there is an additional error to consider associated with the edge movements arising from those which are positively correlated with each other. Since this phase error effectively moves the entire pulse, the error associated with this is also signal dependent. Modelling the phase error as an equivalent error in a PAM system, $\emptyset$ represents the sampling jitter in the impulse train and hence a 'frequency blurring' of the intended signal. Thus the positively correlated error in the pulse edge errors phase modulates the signal slightly. This effect is small, especially for oversampled signals, and if random or at a frequency which places the error outside the audio band is most likely to be inaudible. It becomes most severe when the amplitude of the signal is large (ie. a large amount of energy is moved) so masking is likely to occur, and is band-limited due to the integration of such errors by the nature of PWM (clock deviation that occurs and is cancelled within one pulse will not affect the edge times). A pictorial example of this is shown below.

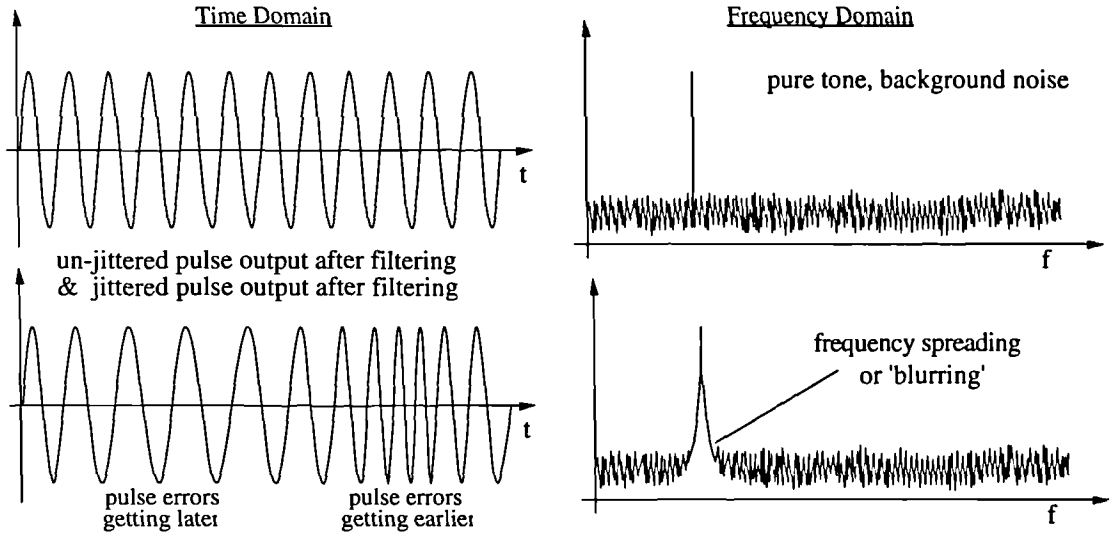


Figure 7.1.3.b : Frequency Blurring Resulting from Phase Error in the PWM Pulse Timing.

A constant size of phase jitter affects different signal frequencies in the audio band with different significance since the frequency deviation is a function of the signal period. The most significant change in frequency occurs for the smallest signal period (ie. the highest signal frequency). Fortunately, this is where the effects are least audible so jitter of this kind is expected to be benign.

For quantisation noise and switching noise to be at comparable levels, equating equations 7.1.3.a & f,

$$\frac{A^2}{12.2^{2(b-1)}} = \frac{L^2.f_n^2.A^2.(\sigma_1^2+\sigma_2^2)}{f_{clk}^2} \qquad \text{Equation 7.1.3.g}$$

† this assumes a one ohm load and should be scaled appropriately if used other than for signal to noise ratio calculations or other comparative measure.

and assuming $f_{clk} \approx 2^b \cdot f_n$ (ie. in the absence of noise shaping), and $\sigma_1^2 \approx \sigma_2^2$ (ie. pulse edges separated by longer durations than the longest period of noise in the master oscillator) yields :

$$\sigma_j = \frac{1}{L\sqrt{6}} \quad , \quad \text{measured in units of the input signal's resolution and sample period product.}$$

Equation 7.1.3.h

From this a required improvement factor of 1/L can clearly be seen, however it should be noted that in interpolated systems, this factor become secondary to the baseband quantisation noise; furthermore, for a system without oversampling or interpolation, the error deviation is more than a factor of two smaller than one LSB time increment. For example, taking the 16 bit 'CD quality' case the ±173 ps assessed previously as the intersymbol interference limit implies the error deviation should be maintained at less than 141 ps for the jitter noise power to be smaller than the quantisation noise power.

Recalling that both circuit related sources and clock related sources make up pulse edge jitter, these two sources will now be looked at for their characteristics and typical sizes, to show how jitter can be minimised.

Time offsets added by circuitry arise from the finite slew rate and gain of amplifier stages, in clock buffering, power switch drivers and logical evaluation of signals. For instance, a logic gate with a threshold instantaneously upset by 50 mV. of noise and a signal slew rate of 2 V/ns, a jitter of 100ps will be introduced. If the transition is modelled as having constant slope near the threshold voltage, then the output jitter in time is proportional to the input noise voltage. This leads to the noise voltage characteristics of the threshold being transferred to the output jitter characteristic (ie. a broadband noise voltage leads to a broadband jitter offset). Recalling that inherent quantisation noise masks the required improvement in pulse edge timing for interpolated signals *if the error is white*, time offsets added by circuitry are not as damaging as might first appear, but as will be seen shortly, these errors are nearly large enough to limit the quality of the system to less than 'CD' quality.

Power supply variations are the main cause of threshold change in logic gates and clock buffers so to avoid this, individual power supply regulation should be used for the output stages (including the last clocked stage itself) and any clock buffer to the last clocked stage. Clocked elements within the counters used in the PWM have almost no influence on these errors since re-clocking of the output can be used to clean up the output signals.

Having separated the vital output and clock stages from other sources of PSU variation, it is worth noting that thermal noise or flicker noise in the gates or individual voltage regulators to these stages will still introduce error. Furthermore, any 'memory' introduced by decoupling capacitors and suchlike will introduce signal dependent error (distortion). These problems are small compared to 16 bit resolution, and insignificant compared to power stage non-idealities so they will not be discussed further, but when higher resolution systems (20 or 24 bit) are attempted they should be considered.

Silicon based logic circuits can achieve edge jitter of around 100 ps and GaAs logic, 10 ps, [GBL91] but power switches and their drivers are extremely difficult to design with such a low jitter specification. The best figure known to date is 100 ps [DEI92] although most commercially available devices are about an order of magnitude worse.

Errors in the clock timing arise from noise entering the feedback loop of the oscillator used to define the clock. Because of the feedback that is applied, an error advancing or retarding the clock will

not only affect the next edge-time but also all subsequent edge-times until a counteracting error occurs. This accumulation of the error changes the characteristic of the clock's power spectral density such that it has a $1/f_m^2$ characteristic (where f_m is the deviation frequency from the ideal spectral line). Further parametric effects introduce a $1/f_m^3$ characteristic very close to the carrier, and broadband noise sets a 'floor' to any oscillator performance. By integrating all these frequency deviations the RMS jitter in time can be calculated as will be shown for the test case below.

Oscillator circuits produce phase noise which can be converted to equivalent frequency deviation in the clock and hence the time jitter over a specified bandwidth. This jitter should be added to the logic and driver jitter (in an RMS sense), satisfying, in total, the overall jitter figure as given by equation 7.1.3.h . Measurements taken from a typical quartz crystal oscillator module suggest that the near-carrier phase noise as a ratio to the carrier power (N/C) is $1/f_m^3$ dominated :

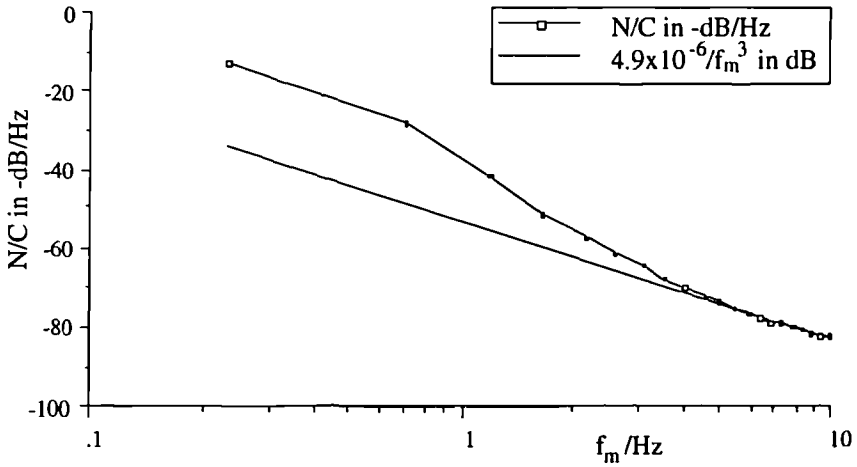


Figure 7.1.3.c : 126 MHz Crystal Oscillator Output Spectrum (narrowband).

From figure 7.1.3.c, the $1/f_m^3$ relationship can be observed in the narrowband phase noise performance and the constant of 4.9×10^{-6} can be derived. It should be noted that significant error is introduced in the measurements at very near carrier frequencies from spectral leakage in the analysis filters, so the low frequency measurements have been ignored. Wideband spectral measurement of the same crystal showed an overall floor of -141 dBC/Hz beyond 500 Hz offset, which masked any transition frequency to a $1/f_m^2$ relationship that might have otherwise been expected from the module's output amplifier. Thus the overall phase noise curve is approximated by :

$$\Delta f^2 = \int_1^B 2.\left\{\frac{4.9 \times 10^{-6}}{f_m^3} + 7.1 \times 10^{-15}\right\}.f_m^2.\,df$$

Equation 7.1.3.i

In the worst case (B = 126 MHz) this yields a total frequency deviation of 168 kHz and hence an RMS jitter of 10.5 ps. From this, the clock jitter can be seen to be considerably smaller than the circuit jitter (unless GaAs logic is used) and it should be remembered that several stages of amplification will be required introducing several sources of circuit jitter (output latching, level shifting, power device drivers and the power switches). However, the slew rates in the circuitry may well be higher than 2V/ns, and at logic level the PSU noise can be kept below 50 mV. with good circuit layout and complementary signal usage (ECL is notably good for this since load current is almost independent of output level).

To assess whether these (approximate) estimates of clock jitter were optimistic, gross jitter

was used to frequency modulate a clock to do a spot check on the PWM output performance. By frequency modulating a stable clock source with band-limited Gaussian noise, the 'clean' and FM'd oscillator signal shown below (figure 7.1.3.d) were used to clock the output of an 8 bit, 8x oversampled DPWM producing alternating asymmetric modulation type from a 16 bit data source using 4 th. order sinusoidal noise shaping.

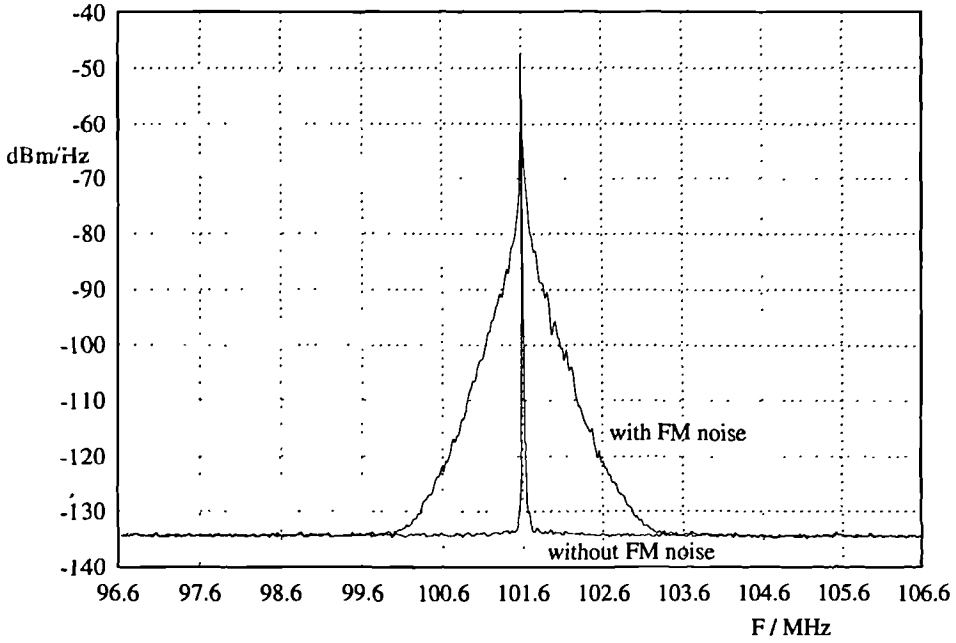


Figure 7.1.3.d : Test Clock Spectrum, With and Without Jittering by Frequency Modulation.

An output spectrum with and without 1 ns jitter and using a 1 kHz -15 dB tone is shown below (figure 7.1.3.e), from which both the characteristics of jitter can be seen : spectral broadening of the tone and wideband degradation of the noise floor. The SNR is reduced from 88 dBFS to 73 dBFS by introducing this amount of jitter (although the jitter level is 17 dB worse than that predicted by equation 7.1.3.h). This reduced degradation compared to theory may be due, in part, to the use of a sinusoidal noise shaper whose residual shaped quantisation noise masks the available resolution before jittering.

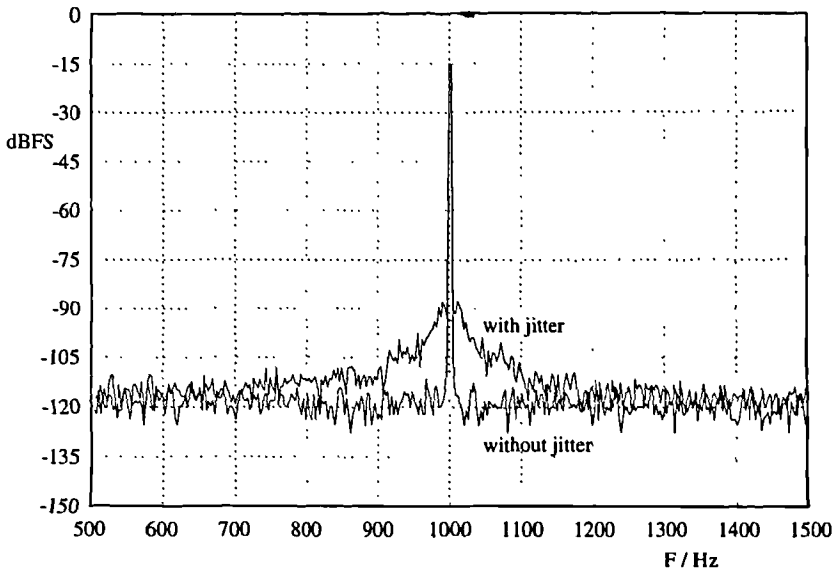


Figure 7.1.3.e : Output Spectrum (narrowband) for a DPWM, Gaussian Jittered to ± 1 ns.

With these same parameters, the influence of clock jitter has been assessed for various sizes of jitter by simulation [WON93]. This is a particularly slow approach since the PWM output has to be interpolated further by the number of levels to which the jitter is quantised. In the example below (figure 7.1.3.f) decimation by a factor of 262144 was required and high resolution FFTs were used to minimise window effects (ie. prevent spectral leakage masking tone broadening). As discussed in section 3.6, comb filters are ideal for speeding up these initial decimation stages.

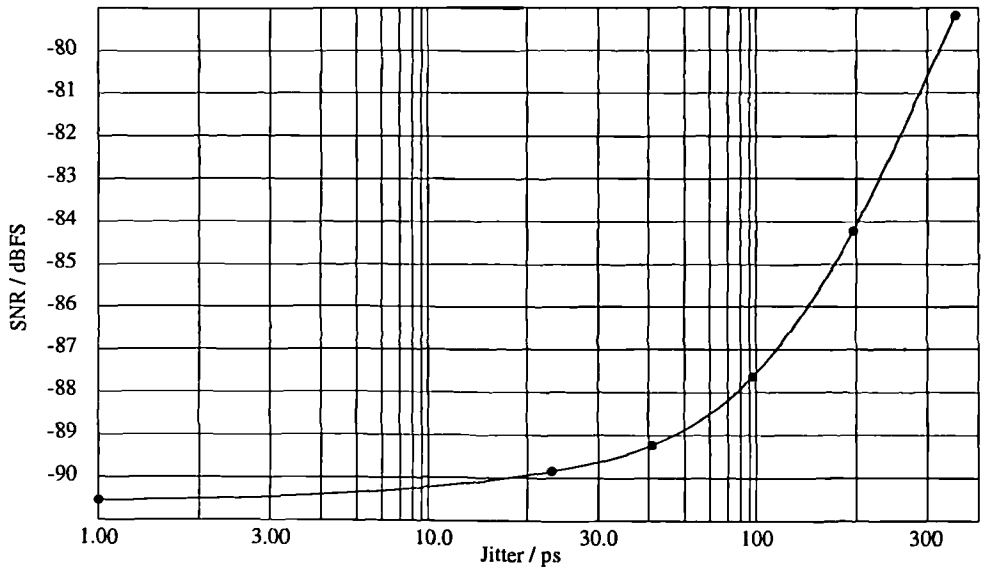


Figure 7.1.3.f : A Summary of SNR In the Presence of Jitter for a DPWM Output.
(nb. the SNR begins to approach a limit for no jitter which is dominated by noise from the noise shaper and the inherent quantisation noise at the input).

In these experiments, the constituents of jitter are approximated to speed up the simulation since the $1/f_m^3$ near-carrier noise response is time consuming to generate. Broadband noise can be generated by simply calling a random number generator but the $1/f_m^2$ noise PSD has to be generated by *accumulating* a random number generator's output to mimic the action of the clock generator.

To summarise, from this series of assessments of the theoretical, simulated and experimental performance of a DPWM signal the limiting factors can be seen to be predominantly from jitter in the output latch and subsequent circuitry, rather than the clock source. Also, the amount of jitter required to significantly degrade a 16 bit, 44.1 kHz sampled signal is circa 140 ps and this can be obtained from silicon based circuits. However for higher resolution signals, lower jitter technologies may be required (eg. GaAs). Lastly the spectral effects of jitter are predominantly spectral broadening, through frequency modulation of the signal, and noise floor degradation as a result of corruption of precisely shaped noise. Both of these effects are thought to be benign since the presence of a large enough signal to introduce noise more significant than the inherent quantisation, will provide more than enough masking to make the jitter noise inaudible in a psychoacoustic assessment.

7.1.4 : Power Supply Stability Requirements

Pulse area can be modified by unexpected amplitude variations leading to a distorted or noisy output after filtering. Since the power supply line inadequacies are directly passed to the output filter, the power supply has to be tightly controlled in class D output stages (more so than in analogue amplifiers where current is controlled).

This implies that simple smoothed rectified AC is not suitable for class D amplifiers and power supply regulation should be used. For high efficiency in the power supply as well as the output switch, a switched mode regulator should be used although this introduces problems of its own which can interfere with the output devices themselves. Feedback of the line borne switching noise can quickly upset the accuracy of the control loop in the regulator and substantial filtering is required to reduce this.

For audio use the pulse area needs to be controlled to levels with an error of 1 part in 2^b, typically 100 dB., so very tight amplitude control is required although uncorrelated noise at higher frequencies can be less tightly defined so long as correlated AM -> PM conversion does not become introduced as a result.

Switching regulators are usually operated at several tens of kilohertz and care has to be taken to ensure that the regulator switching frequency does not 'beat' with the PRR to introduce audible tones. This can be done by heavy filtering to totally suppress the regulator output, or by locking the regulator to a subdivision of the PRR.

For the regulator to appropriately control the power supply level, the control circuit in the regulator must have a response time substantially higher than the audio bandwidth. Typically 40 to 50 kHz should be used which suggests that the regulator operates at 88200 Hz or 176400 Hz for say and eight times oversampled system (PRR=352800 Hz).

7.2 : Circuit Problems and their Effects.-

7.2.1 : Overview

When building an output switch and filter for a class D amplifier the theoretical performance evaluated in the last section has to be maintained through each part of the driver and switching circuits. Generally, this performance becomes more difficult to achieve as the power level increases and the frequency of the output pulses (and their edges) increases. This is as a result of parasitic effects such as package inductance, stray capacitance, delays required for device commutation and layout-sensitive problems such as 'ground bounce' and crosstalk.

The basic operation of the output switch for a class AD power stage involves using two MOSFET devices operated in a complementary fashion, to connect either the positive or negative power rail to the load via a low pass filter (see figure 7.2.1 below). The timing of the two devices is crucial to good performance and efficiency. It is important that they are not 'on' at the same time (and hence shorting the power supply - 'shoot through') and yet each MOSFET should connect the required power rail to the load as near as possible to the required time.

Having an output filter directly after the power switches forces the first arm of the filter to be inductive to avoid large in-rush currents at switching time (which would be the case if a capacitor were the first filter component). This forces the load to appear inductive to the power switches, so to avoid damaging flyback voltages ($-L.\partial I/\partial t$) recirculating diodes are used. These put large voltage spikes into the rails which require substantial capacitance to ensure the power rail level is not significantly disturbed. Thus the basic switching circuit and filter is as shown below :

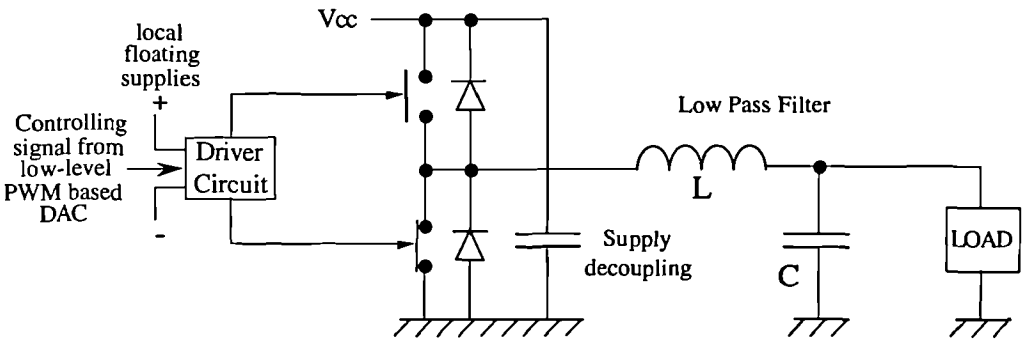


Figure 7.2.1.a : Basic Output Switch Schematic.

To change the output power level the voltage to which each pulse is switched can be controlled (V_{cc} as shown above). This enables finer power control than by pre-scaling in DSP and avoids requiring greater dynamic range in the DAC overall (this would otherwise have to be increased by a further 4 to 6 bits to around 20-22 bits [CRA92]). Problems exist in the driver circuits if a variable positive rail is used in this way, since the gates of the two switching devices have to be referenced to the rail and thus the gate separation has to become variable.

Independent voltage control but precisely aligned time control of the upper and lower devices is the main problem for the drive circuits' design. Allowing for differences between the characteristics of the two devices is the second most significant issue. These will be discussed next.

7.2.2 : Switch Timing Problems

Numerous problems exist within the output switch, causing asymmetry in the delay to each edge's transition, the edge's shape or the edge slew rate. These arise mainly from three sources : the difference between the driver circuits' propagation times (t_{plh} and t_{phl}), the differences in the power MOSFETs input capacitance (C_{iss}), and the differences in the power MOSFETs feedback capacitance (C_{rss}). The driver circuit problems can typically be dealt with in the same way as the power stage's problems since the differences in the drivers arise from the similar problems of variations between P & N-channel MOSFETs although on a lesser scale.

In a typical driver stage, complementary P & N-channel devices of circa 5A peak current handling capacity are used to drive the gate of each MOSFET. By using complementary drivers, the drive voltage applied to the MOSFET gate can be adjusted until the driving device switching thresholds occur at the same voltage allowing fastest possible switching of the complementary pair (on the border of shoot through). This avoids complicating the driver which tends to introduce layout problems with the associated additional complexity required to provide dead time in the switching to avoid shoot through by other means. The choice of devices for adequate speed and peak current handling capacity (eg. VN10LM, VP0610L) yields threshold voltages of circa ±4v which then requires a drive voltage of 8v, sufficient to drive the power device into a fully switched state. A subsection of such a circuit is shown below :

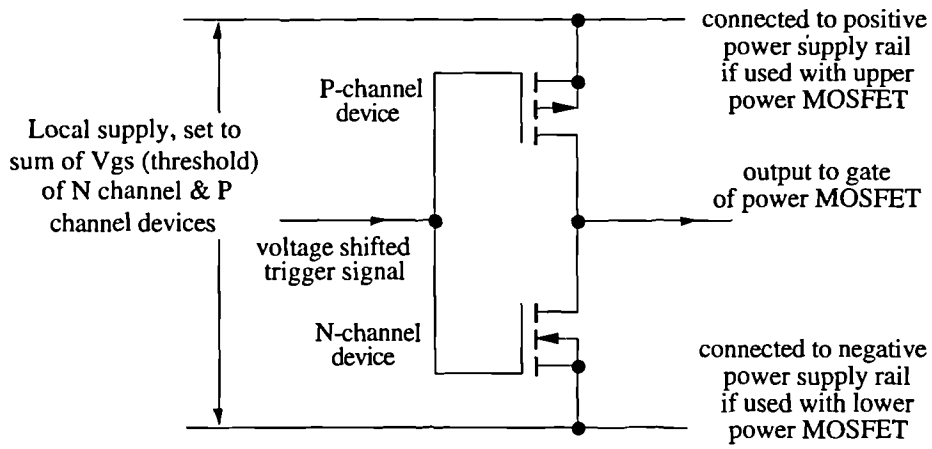


Figure 7.2.2.a : Minimum Switching time Complementary DMOS Driver Circuit.

Unfortunately, since variable supply rail separation is required for output power level control, this technique cannot be used to with the power devices themselves. Since the channel resistance of P-channel devices is higher than N-channel devices larger devices have to be used in the 'high side' (attached to the positive rail) if complementary switching is required. This implies the high side driver will see a higher load capacitance and thus will be slower. To correct for this, the low side driver can either be slowed down by adding further load capacitance or the high side driver can use two MOSFETs in parallel to increase its drive current handling capacity (MOSFETs exhibit positive resistance variation with temperature and thus will current share evenly). The parallel circuit itself will have a higher input capacitance so this is only a referral of the problem rather than a cure, but the problem

becomes progressively smaller each time it is referred back into the driver chain. In experimental circuits this proved more useful in the driver stages than in the power device itself; a circuit of this is shown below:

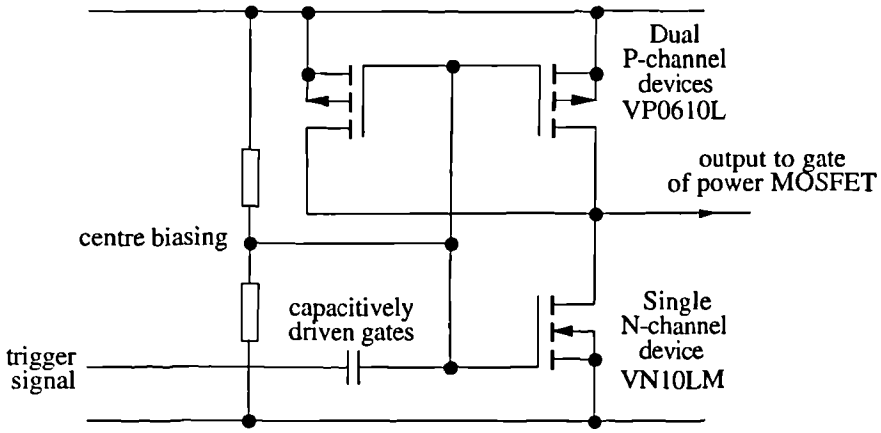


Figure 7.2.2.b : Parallelled P-channel Devices to Reduce Drive Current Asymmetry.

Instead of complementary P & N-channel output stage, two N-channel power MOSFETs can be used to ensure similar drain resistance and hence similar drive requirements. In this configuration the high side driver will use output voltages above the positive power rail to switch 'on' the high side device which requires additional circuit complexity. Also, the entire drive circuit with its floating supply will have to move up and down with the source voltage (ie. with the output). Small stray capacitances between this entire circuit and low side voltages yield this configuration unusable since the high device gate threshold will not be properly controlled. For this reason this configuration will not be considered further.

Since the main problem with persuading the output devices to turn on and off quickly is charging the gate capacitance, large currents and adequate drive voltage must be made available, to force charge into the device. Large drive voltages can breakdown the gate insulation and must not be maintained above the gate-source breakdown voltage ($V_{gs(max.)}$). Since the package lead inductance limits the rate of change of current into the device short peaks of voltage can be applied, exceeding $V_{gs(max.)}$ to push adequate charge into the device in the required time provided the drive voltage is removed once adequate charge has been applied. Some manufacturers supply gate charge information for calculation and this is of the order of 15 nC. (@ 10 V for the IRF510). Thus, for example, a ten volt gate transition (from fully off to fully on) should be possible inside five nanoseconds if a current of three amps is used. The VN10LM is quoted at having a low $R_{ds(on)}$ (3Ω at V_{gs} circa 10v) hence the choice of this device. The $R_{ds(on)}$ of the p-channel device (VP0610L) is quoted at twice that of the N-channel device hence the choice of two parallel devices as shown in figure 7.2.2.b .

Unfortunately under rapid turn off conditions, the parasitic anti-parallel diode inherent to D-MOSFET devices can start to conduct, and this has a very slow reverse recovery characteristic (65 ns. for the IRF510 device). To avoid this, faster diodes have to be placed outside the device to prevent conduction of this parasitic affecting the load. This is also necessary since when driving inductive loads, the flyback voltage at the end of a current pulse should be recycled through the power supply rather than allowing a pulse error at the end of each edge. The anti-parallel diodes outside the MOSFET

are not adequate on their own since the parasitic diode may still begin to conduct while the flyback diodes are in conduction. To avoid this the flyback diodes can be offset, so that the flyback is limited to V_{cc} and ground rather than $V_{cc}+V_{diode}$ or ground+V_{diode}. This has the additional benefit that when the output is driven to predominantly positive or negative voltages, it will still remain symmetric about zero, avoiding 'crossover distortion', but it has the drawback that efficiency may be reduced since the recirculating energy must now be dissipated instead of reused. Offset diode positions are shown below.

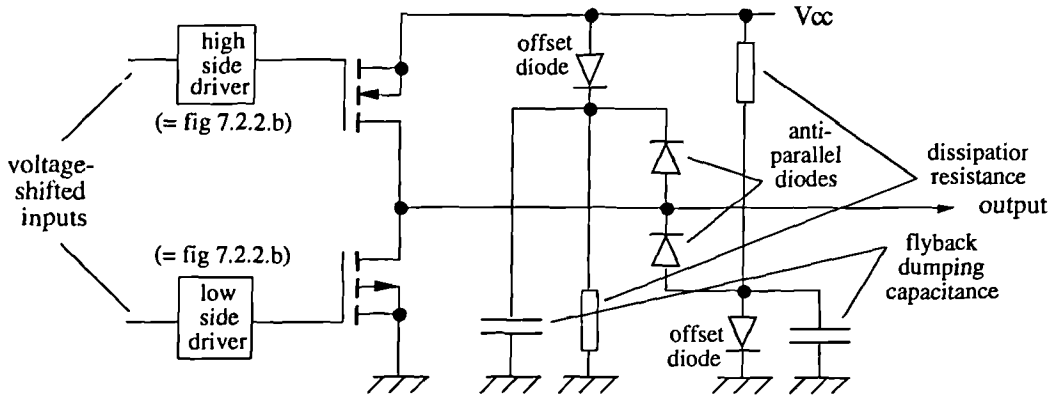


Figure 7.2.2.c : Anti-Parallel Diodes to Prevent Parasitic Diode Conduction in the MOSFET

To maintain fast capture of the flyback energy and avoid reverse recovery time, Schottky diodes should be used such as the MBR150 (V_f = 0.2v). It should be remembered that when the anti-parallel diodes are conducting they pass the full current that the MOSFET was handling at turn off, thus if both MOSFETs are allowed to turn on simultaneously for a short period of shoot-through (by mistake or design variation with temperature, etc.) the Schottky diode may well be ruptured since these do not have the positive temperature coefficient of resistance to protect them that the MOSFET does. The effects of the modifications of figure 7.2.2.c are shown below for a rising pulse edge:

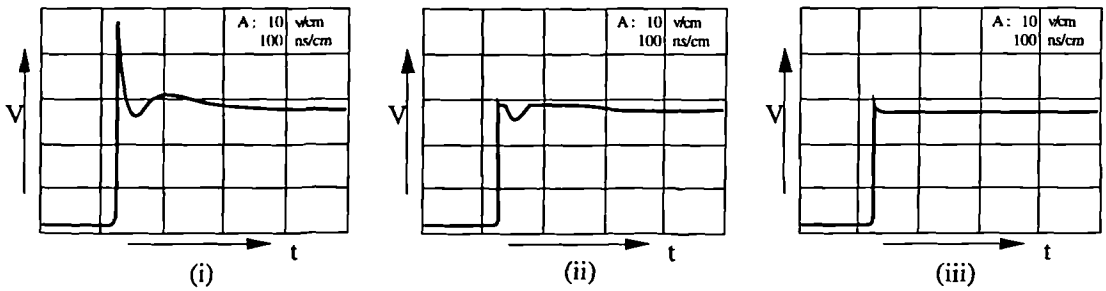


Figure 7.2.2.d : Output (i) No Diodes, (ii) Anti-parallel Diodes only, (iii) Both (ii) and Offset Diodes.

In a similar way, if asymmetries exist between the $V_{ds(on)}$ for the high side and the low side arising from the different $R_{ds(on)}$ for a similar load current, the output voltage will again exhibit crossover distortion. This is more difficult to eliminate since the voltage is load dependent and hence the current varies with the signal and load, so V_{ds} cannot be cancelled except under steady state conditions (by altering the offset voltage used for the flyback dumping capacitors). The simplest solution is to reduce $R_{ds(on)}$ by appropriate selection of output devices (eg. IRF510's $R_{ds(on)}$ @ V_{gs}=10v is 0.4Ω †) and to drive the gates to completely turn on the device whenever active.

† the 101N30-00 from Directed Energy, Inc. is capable of $R_{ds(on)}$ as low as 0.05Ω.

7.2.3 : Supply Impedance

As mentioned in section 7.1, variation in the power supply voltage introduces pulse amplitude modulation. This is usually of the form of noise, regulator switching frequency or mains frequency feed-through. Tight amplitude control by a high gain in the PSU control loop can be used to reduce this to acceptable levels.

Since the load may require pulsed currents of tens of amps, supply impedance is of great concern because slight voltage droop in the level of what is assumed to be an ideal voltage source leads to differential linearity error particularly at high and low filtered output voltages. This introduces significant odd order harmonic distortion and should be avoided.

One technique for reducing this effect is to reduce the power supply impedance as much as possible by using wide tracks, adding large (local) capacitance and using advanced control within the regulator (ie. a band-limited PID control strategy).

The second technique is to use the MOSFET sources as the feedback reference voltage for the PSU; high impedance (low current) measurement of the actual supply voltage can then be used. Sampling of the MOSFET drains during their 'on' part of the cycle can also be used to reduce the effects of the differences in $R_{ds(on)}$; if such sampling is employed, this should be controlled such that it is regularly timed so that after A/D conversion, digital low pass filtering can be used to avoid aliasing from the switching (PRR) components (ie. only the audio band should be compensated for).

A third possible technique for reducing supply droop is to pre-compensate for it in the DSP by expanding the signal which will be subsequently be companded by the switch (this will be complicated by PWM's non-LTI behaviour but may suffice to a first order approximation).

7.2.4 : Switching Interference

The output switch handles large rates of change of current which broadcast electromagnetic interference. This can be a nuisance to neighbouring devices (amplifiers are often placed near to radio receivers!). Output filtering greatly helps to reduce the amount of high frequency power escaping by conduction through the load terminals but radiated interference is more difficult to control. Conventional shielding is required to separate the recirculating currents (power switch, filter and supply decoupling capacitors) from all others in the amplifier so that parasitic feedback is also kept to a minimum (the amplifier can be a nuisance to itself if radiated power is allowed to upset the control loop of the power supply regulator for example). Secondary shielding should also be used to separate the entire amplifier from other circuits and if possible the amplifier should be kept away from susceptible equipment since field strength falls rapidly with separation.

The problem of conducted and broadcast EMI is so great that the amplifier may be best suited to active speaker applications where the signal could be taken in optically, and the amplifier, its supply, its modulator circuits, and the output filter and load could all be in one shielded environment.

7.3 : Circuit Configurations and Performance.

7.3.1 : Overview

As discussed in the last section, switch operation can be performed by a complementary pair of power MOSFET devices each driven by complementary pair of low level DMOS devices. Two main alternative configurations exist for the arrangement of the power MOSFETs: 'Single-ended', where the load is driven from one end only and the load current is returned through the ground rail, and 'Bridged', where two pairs of output devices are used to drive each end of the load in opposite directions without use of the ground rail. Driving only one end of the load simplifies the output circuitry by a factor of two but also halves the voltage applied to the load; this is commonly called a 'half-bridge' configuration. Driving both ends of the load in opposite directions (anti-phase) is called the full or 'H-bridge' configuration. In advanced designs these two ends need not be strictly anti-phase. They can be controlled separately using separate supply levels (eg. for fine and coarse quantisation), or sliding between similar and complementary signals for 'volume control' by subtraction (ie. varying the proportion of common-mode and differential-mode signal).

In this section, basic configurations for the half and full bridge topologies will be evaluated, assuming identical complementary drivers (two or four as required). Circuits suitable for shifting the logic-level input to the floating-driver level using capacitive and optical coupling will be presented, and pre-driver circuits for boosting the logic-level signal and producing exact complementary signals will be looked at. One example of a post-driver circuit to boost turn-on and turn-off speed will also be examined. These can be combined as shown below, but will be described in isolation since their interactions are very layout dependent and hence implementation dependent.

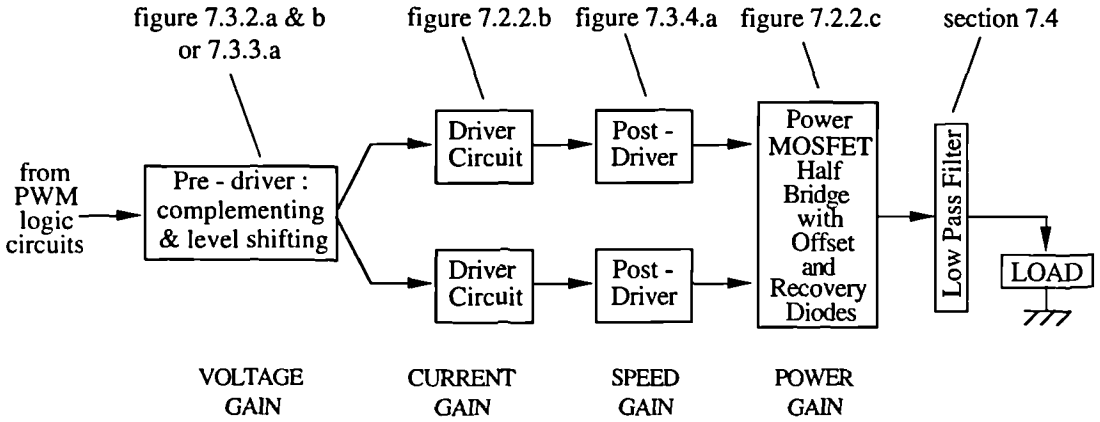


Figure 7.3.1.a : Half-Bridge, Output-Circuit, Block-Diagram.

Finally, measurement from a 2W version of an output stage will be discussed, illustrating the benefits, in distortion terms, of using the full bridge configuration.

7.3.2 : Pre-driver Circuits

Although the basic gate drive circuit has already been discussed interfacing to logic-level devices is required and more advanced designs can be used for fixed output voltage levels.

For the full bridge circuit, complementary signals are required with very precise time alignment. This can be achieved using a long tailed pair arrangement (differential amplifier) followed by totem pole output stages. The long tailed pair operates by a balance of emitter currents (bipolar devices are required for this). In order to improve the operation of the circuit, a MOSFET based constant current sink was used in place of the 'long tail' and this also avoided the use of further (large) power supplies. Since high speed operation was required, bipolar transistors with high f_T were used (2N5109, 2N5583) and low collector resistor values. Fish beads (ferrite) were added to the constant current sink to help reduce wiring inductance inducing changes in the tail current. The circuit is as shown below :

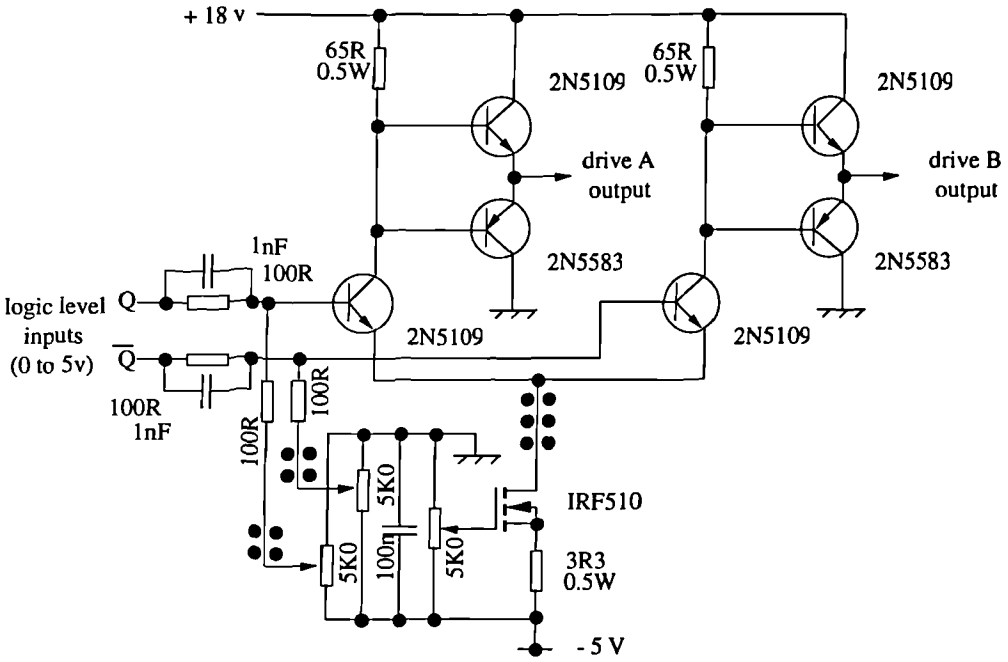


Figure 7.3.2.a : Long Tailed Pair Complementer

With a varying supply separation the above circuit could not be used on its own since at low supply voltages the current source can no longer stabilise the circuit properly. An additional circuit was used, employing a voltage amplifier from the output to produce a high amplitude version of the signal followed by capacitively coupling this into both the high-side and low-side driver circuits as shown in figure 7.2.2.b. The voltage amplifier used a similar strategy to the driver circuits of doubling up on the number of p-channel devices, but these are now used as an active load. Bias for the active load was provided from a local reference (Zener diode) and by inserting inductance in the source of the top devices, the output signal could be encouraged to overshoot to assist with fast gate charging in the next stage without causing gate damage; the circuit schematic for this is shown overleaf:

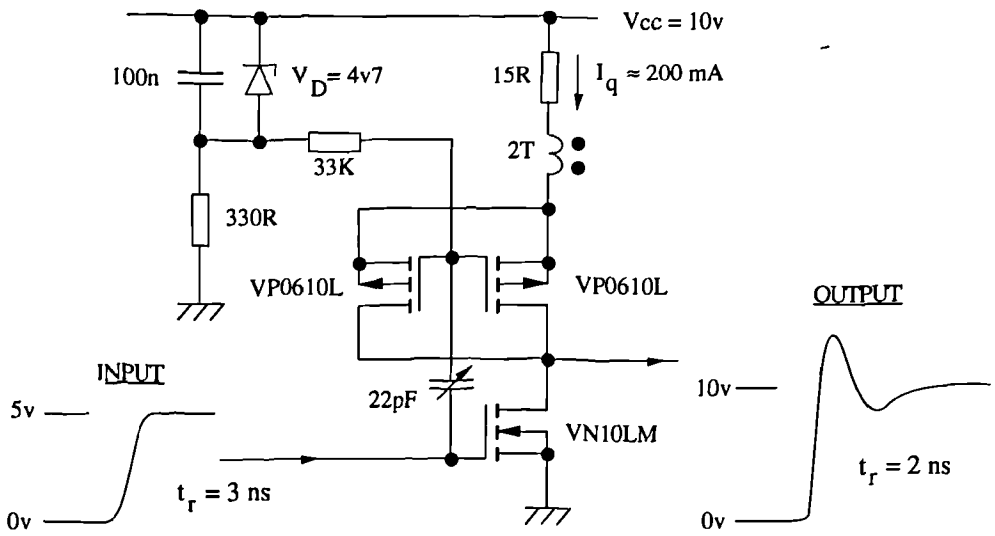


Figure 7.3.2.b : Active Load Voltage Amplifier with Controlled Overshoot.

7.3.3 : Optical Isolation Level Shifting Circuit

Level shifting as shown in figure 7.2.2.b using a capacitor to store the voltage offset between the amplified logic signal and the driver circuit input causes signal levels to be affected under large signal conditions and power rail noise. Since the offset is set by the bias resistors in the driver circuit, which attempt to maintain the driver input voltage near the middle of the driver rails, the capacitor will become discharged for high duty-cycle drive pulses and overcharged for low duty cycle drive pulses. By choosing the quiescent drive pulses to be 50% duty cycle and knowing the signal source to be alternating with frequencies greater than say 35 Hz, the time constant of the RC pair can be made long enough for the switch to operate satisfactorily. Fortunately the input resistance of MOSFETs is high and hence does not interfere with biasing; for tests, both R_{bias} were chosen to be 810 kΩ and C_{shift} was chosen to be 100 nF., giving a time constant of $\tau = 0.028$ s (one half life).

Using this long time constant allows the circuit to operate without causing both driver devices to turn on simultaneously under large modulation depths, but requires the drive circuits to operate from different parts of the driving signal's edge. Because the rising edge of the amplified logic signal (the output of figure 7.3.2.b) is not instantaneous, operating on different parts of the rising edge for different duty cycles introduces pulse shortening for predominantly positive parts of the audio signal and pulse lengthening for predominantly negative parts. This manifests itself as even order harmonic distortion which is more acute at higher modulation depths.

Offset voltage variation (and the associated distortion) can be reduced by using 'dumping diodes' to clamp the driver's input voltage to between its power rails. Thus if the level begins to shift, the offset capacitor is injected with a pulse of charge from the rail to maintain the offset. This is only useful if the input signal swing is similar to the driver's supply rail separation and prevents the action of the controlled overshoot as discussed in the last section. Bootstrapping of the level shifter or current mirroring of the low side driver are also possible circuits, but operating these with several hundreds of megahertz bandwidth is not easy.

An alternative approach to the level shifting is to electrically isolate the input from the driver using a pulse transformer or optical isolator. Unfortunately the pulse transformer also suffers from modulation depth limitations, so an optical isolation stage was designed. Clearly the bandwidth of the optical isolation has to be very high so the fastest available device was used (6N137). Although this does not have the same bandwidth as the logic-level PWM signal the output can be amplified and 'squared up' although this does introduce time jitter. A design based on this is shown below :

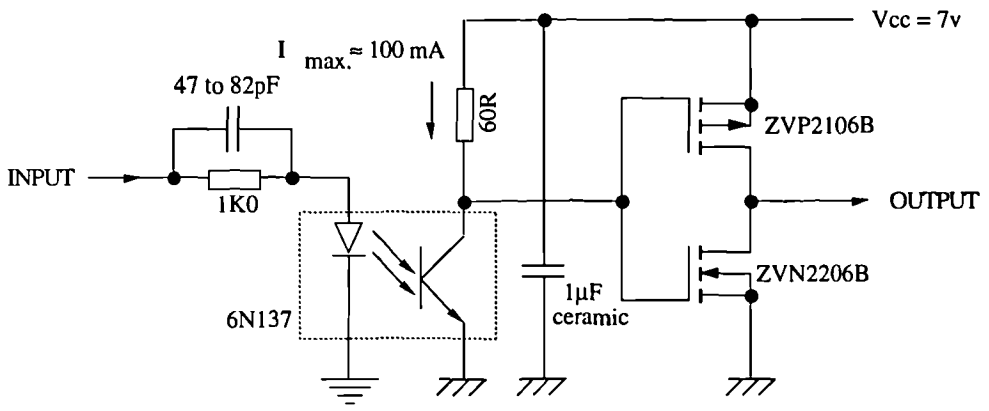


Figure 7.3.3.a : Basic Optically Isolated Driver.

Since the LED stage of the optical isolation is capacitive, a speed up circuit was added to the input, and for fast operation of the output the lowest possible collector resistor was used. Unfortunately, asymmetry in the rising and falling edges is large and has to be compensated for subsequently. Also, jitter in the optical connection was found to be unsatisfactory for operation beyond about 14 bits. To avoid this the signal should be resynchronised using an output stage as described in section 6.3.5 and subsequently amplified up for the driver circuits and then the gates. Because the resynchroniser requires a clock, the fast counter-clock has to be distributed to all the drive blocks which complicates the design, but because this signal is of 50% duty cycle, it does not vary with the PWM audio duty cycle. Care is required to ensure that bringing the fast clock near the large currents of the drivers and output stages does not introduce interference into the clock line since this is the primary accuracy of the entire process; shielding of the clock and appropriate physical separation should be used.

Signal timing can be maintained with better accuracy if constant duty-cycle pulses are transmitted across the level-shifting interface. This then allows the use of pulse transformers (avoiding opto-coupler jitter) but adds additional complexity. Typically, 'set' and 'reset' pulses are sent to the driver circuit, which should now include logic to regenerate the output pulse shape using the fast counter clock for resynchronisation. Field restoration is required in the transformers so the set and reset signals should be bi-phase; also these signals should be completed within the minimum pulse duration time (or the minimum time between pulses), and hence should be short (typically 100 ns). Such additional complexity only becomes necessary for very high power amplifiers and is remains the subject of future research.

7.3.4 : Post Driver Example : Snap Diode Driver

For very fast rise and fall times the following circuit [SIL89] can be added after the gate drive circuits (figure 7.2.2.b) and in test was capable of 300 ps rise times and 1 ns fall times.

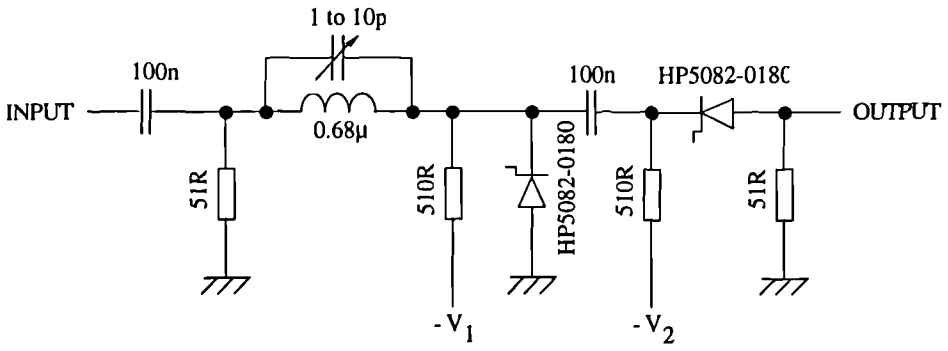


Figure 7.3.4.a : Snap Diode Charge Dumper

The circuit operates by storing charge on a snap diode (varactor) which when removed will suddenly go from a low to a high impedance state. As the varactor voltage is discharged the potential difference from its charging source increases smoothly (and with it the discharging current) until the varactor 'blocks' as it goes to its high impedance state. By discharging the varactor through an inductor the abruptly stopping current induces a voltage spike with a very fast rise time (typically tens of picoseconds). This can be passed to the gate of a MOSFET to charge it very quickly. The amount of charge transferred is dominated by that taken from the varactor and can thus be well controlled. By passing the required charge for turning the MOSFET gate full-'on' the internal gate voltage of the device does not exceed the V_{gs} breakdown voltage. Outside the device and its lead inductance, there can be voltages up to ten times the gate-source breakdown voltage being temporarily applied. Using one varactor for the rising edge and one for the falling edge allows rapid speed up of both switching points in the cycle.

Under test the delay from the initiating edge was circa 40 ns and exhibited more jitter than that inherent to the MOSFET itself when measured at slower edge rates. Although this circuit shows potential for high speed driving, it has not been included in preliminary designs and has been left as an area for further research. It is interesting to note that edge rise-time or fall-time is not as important as edge timing accuracy, provided adjacent edges can be prevented from interfering with each other (typically by the use of guard bands).

7.3.5 : Half Bridge & Full-Bridge Output Circuits

The half-bridge circuit has already been described and is shown in figure 7.2.2.c with its associated output diodes. Supply decoupling (1000 µF. electrolytic, 10 µF. tantalum and 3x 100 nF ceramic) was added as close to the MOSFET pair as possible which were chosen to be the IRF510 and IRF9520 (N and P-channel devices which exhibit similar $R_{ds(on)}$ figures and very small C_{iss}). Output

filtering as described in the next section was included to complete the output stage. The power devices were soldered together tab to tab to reduce output impedance and maintain temperature tracking despite differences in $R_{ds(on)}$s (with low temperature solder to avoid dopant migration) but a heat sink was found unnecessary. Separate power supplies were used to provide the load current, the high side drive current, and the low side drive current, and additional decoupling of the drivers was added adjacent to the driver MOSFET pair (10 nF. & 1 µF. ceramic). Operating from circa 30 v. output rails with an asymmetric modulator at 88% depth, driving an 8 ohm load with a -15 dBFS, 1 kHz signal, spectra such as the following were measured (the theoretical distortion level would be below the noise floor).

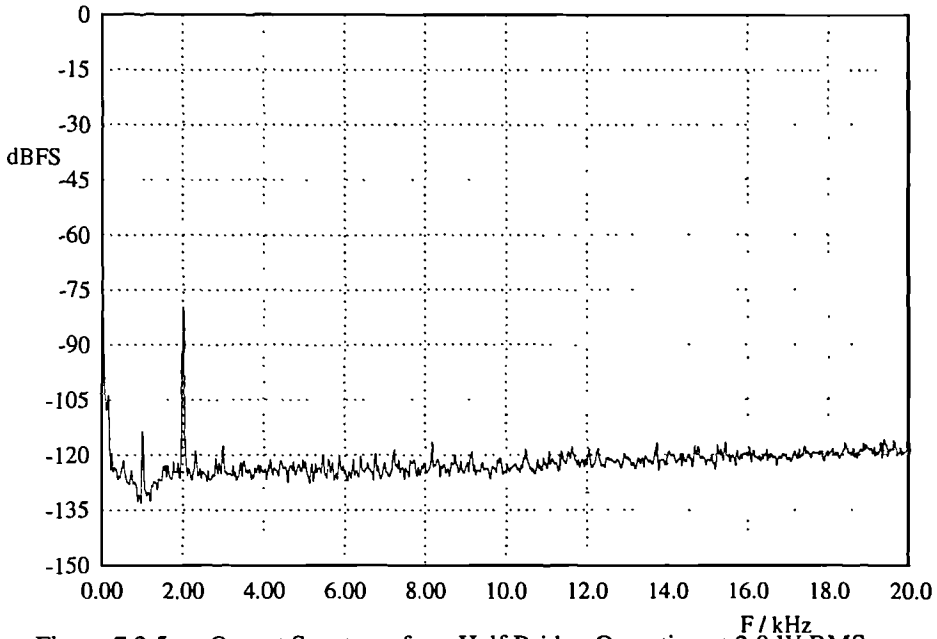


Figure 7.3.5.a : Output Spectrum for a Half Bridge Operating at 2.0 W RMS

In this spectrum, the basic driver circuits can be seen to have successfully turned 'on' and 'off' the power devices with little additional distortion (the fundamental at 1 KHz has been notched out to permit measurement). At higher power levels, the harmonic distortion increases as a result of power supply droop, masking the successful operation of the power switch. Compensation for this is required as mentioned in section 7.2.3, but the basic circuitry can be seen to have demonstrated that power D/A conversion can be achieved using a class D output stage.

By operating two such half bridge circuits in anti-phase and driving the devices with a full scale tone the output power can theoretically be increased to 60 W (limited by the chosen MOSFET breakdown voltages and channel-current handling capacity). Choosing devices capable of higher current handling capability, higher voltage capabilities and faster switching times should allow extremely large amplifiers to be constructed with some ease. The full bridge topology can be made up as shown overleaf.

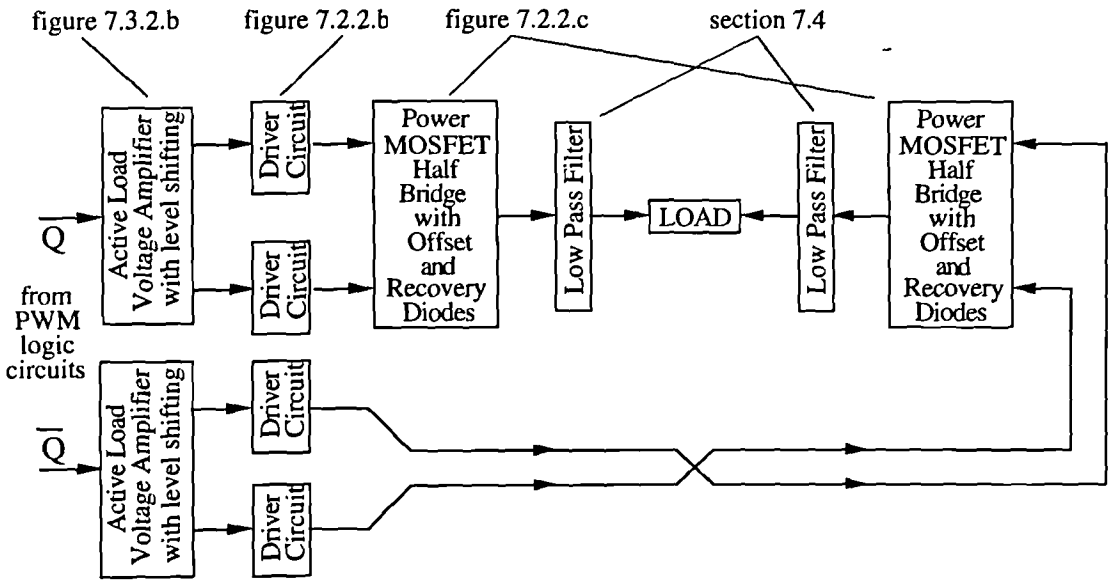


Figure 7.3.5.b : H-Bridge Output Circuit Schematic.

7.4 : Output Filter Design and Performance.

7.4.1 : Filter Requirements

Three output filter applications will be examined, each of which requires a separate design. These are : (1) the measurement of the output for specification, (2) the use of a switched output to drive a loudspeaker in the conventional way, possibly through long wires near a radio or television receiver, and (3) use of a switched output to drive a loudspeaker within the same screened environment as found in "active" speakers.

Measurement of the output using any sampling technique requires an anti-aliasing filter to suppress the carrier, its harmonics and sidebands, prior to A/D conversion. High dynamic range measurements also require the input signal to be removed, but filtering for this is commonly provided within test equipment so it need not be designed separately (eg. Audio Precision's "System One"). Taking an eight times oversampled system (PRR=352.8 kHz) as an example, the signal band should be passed to within ± 0.5 dB from 0-20 kHz, and the first carrier component should be suppressed to below -120 dBFS. Ideally, the passband would be linear phase and flat. In practise, the passband flatness and phase linearity have to be compromised slightly to achieve the carrier suppression. A Butterworth characteristic provides a good compromise between these two [FDI86] where the passband is over-designed in width by a factor of 2.5. By using a reduced signal passband of 15 kHz, and a half power point cut-off frequency (-3 dB point) at 2.5 times this, a 6 th. order filter with Fc=35 kHz will ensure the carrier suppressed by 120 dB (20 dB/decade/order for one decade).

An odd order passive filter was chosen to minimise the use of inductors (the less ideal components) and symmetric input and output impedances were designed for so that it could be constructed with similar valued components and hence minimal component mismatch. This unfortunately forces the first filter component to be capacitive, reducing the filter's input impedance at high frequency, but since the filter was intended for use at low power this is a minor drawback. Moving to the next higher odd order (7 th.) also permitted a wider passband to be designed for (-3 dB point @ ≈50 kHz) an input and output impedance near 50Ω to match measuring systems' inputs, and components near standard "off-the-shelf" values. By selecting capacitors with low ESR, the dominant non-idealities are the inductor series resistance which can be modelled easily using complex nodal analysis, and the saturation of inductor cores which can be avoided by potential-dividing the input with a resistor network.

A similar compromise of passband flatness and phase linearity with complexity of filter and selection of component values was applied to the other two design requirements. A summary of the design specifications arrived at by this process is shown below:

Name	Order	Fc	Gain @ PRR	Gain @ 20 kHz	1st Component	Load Z
Design 1	7	50 kHz	-118 dB	-0.29 dB	capacitive	47 Ω
Design 2	5	52 kHz	-83 dB	+0.18 dB	inductive	8.3 Ω
Design 3	2	85 kHz	-25 dB	-0.02 dB	inductive	8.0 Ω

Specific designs which meet these specifications will be presented next with their spectral performance.

7.4.2 : Analogue Filter Assessment

In order to design analogue filters, two approaches can be taken. The first is to take a standard filter shape such as the Butterworth, Bessel or Chebychev I & II designs and scale and transform it for the application required. These classical solutions tend towards a rolloff of 20 dB per order per decade (6 dB per order per octave) so an order estimate can quickly be obtained with reasonable accuracy. From this the standard form of the transfer function or tabulated coefficients can be taken, using scaling for the load impedance and a simple transformation for the high pass, band pass and band stop designs.

The second approach is to use a non-classical design, tailoring it for a particular application. The combination of passband ripple, stopband suppression and phase or group delay response can be modified or the deviations of real component values from the ideal can be accounted for whether these arise from selecting from a discrete set or from their parasitic elements.

Both these approaches were used in the design of low pass designs for each output filter, starting with a 'classical design', adjusting for the nearest component values, and then calculating the new response (using 'real' components which included the dominant parasitics) through complex nodal analysis for the estimated phase and gain performance.

7.4.3 : Design One Solution

As discussed in section 7.4.1, this design objective is met with a seventh order, approximately Butterworth symmetric low pass filter, matched to 50Ω at both input and output. Ferrite cored inductors were used to reduce space and to approximate the Butterworth characteristics closely (fewer turns lead to smaller parasitic component values : less inter-turn capacitance and smaller series resistance). Similarly, high quality capacitors were chosen for their low ESR and small tolerance to avoid component mismatch which would spoil the Butterworth characteristics. Also, by using symmetry in the structure the components become more easily matched as they are similar-valued.

The implementation strays from the ideal component values mainly through the series resistance of the inductors. After calculating the ideal component values, calculating the nearest capacitance and inductance values available from standard values, and including the parasitic components in analysis, the following network was assessed for its response :

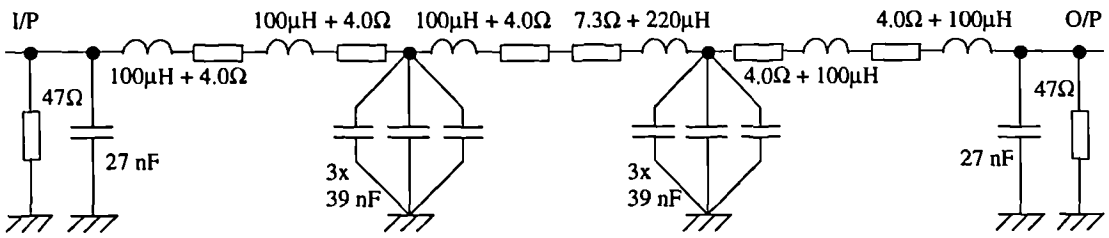


Figure 7.4.3.a : 7 th. Order, Symmetric, Low-Pass Filter Schematic, Including Significant Parasitics.

The frequency and phase response for this filter were calculated and are shown overleaf (figure 7.4.3.b).

These have been compared with measured responses, and are so similar that the plot below can be considered the same as the measured response (see figure 6.3.5.a).

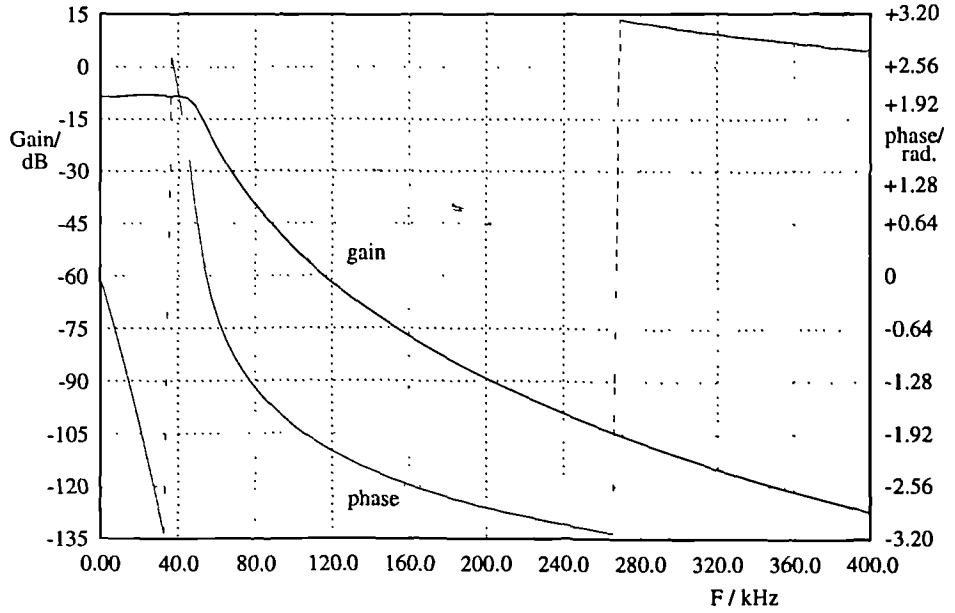


Figure 7.4.3.b : Design One (Simulated) Magnitude & Phase Responses.

This filter was used for most of the output performance plots as shown in chapter 6 and as can be seen above, meets the design requirements of nearly linear passband phase response, nearly flat passband magnitude response as well as the carrier suppression of at least 120 dB.

Since inductors capable of saturating were employed (ferrite cored), the final filter was checked for distortion for several input levels, and signals above 1.25v pk-pk were found to cause distortion at levels which would interfere with results (see below).

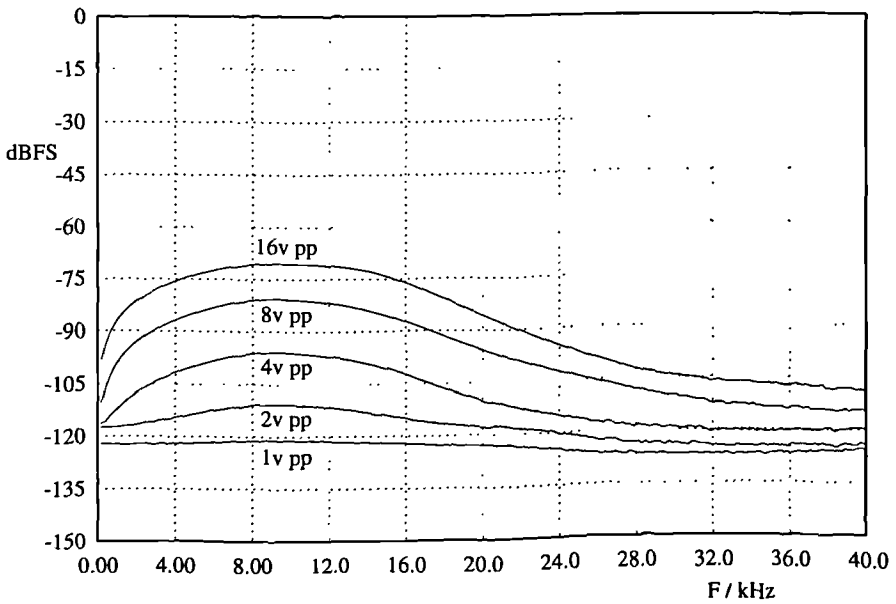


Figure 7.4.3.c : Design One Measured Distortion for Various Input Levels.

To ensure saturation did not happen in normal use, an additional series resistor of 270Ω was inserted at

the input, reducing the filter's input voltage level by between 6x and 7x. Thus, TTL driving voltages could be used with no danger of the measurement filter distorting results; this also reduces the chance of overloading the driving circuits which might otherwise contribute to misleading distortion measurements. This increases the insertion loss to nearly 25 dB which was accommodated for in the final measurements. The input impedance at 1 kHz was measured at 298Ω and a balancing resistor of 300Ω (330R || 3K3) was used in the other complimentary line from the PWM for current matching during tests.

7.4.4 : Design Two Solution

The basic circuit to meet the requirements of this design is that of a fifth order approximately Butterworth low pass filter. The filter is intended for use with an 8 Ω load (ie. at high power levels) so air cored inductors are used to avoid saturation and the first input component is chosen to be an inductor. This choice ensures both an increasing load impedance with frequency for the power switch and an inductor as the last component. This inductance is large compared to a loudspeaker's inductive non-ideality and hence slight load inductance variation (with current/temperature or acoustic load) can be ignored. The increasing input impedance with frequency helps maintain high efficiency in the power switch and correct timing of its edges for high accuracy conversion as a result of reducing the rate of change of current (∂I/∂t) in the switches themselves. The ideal schematic is shown below :

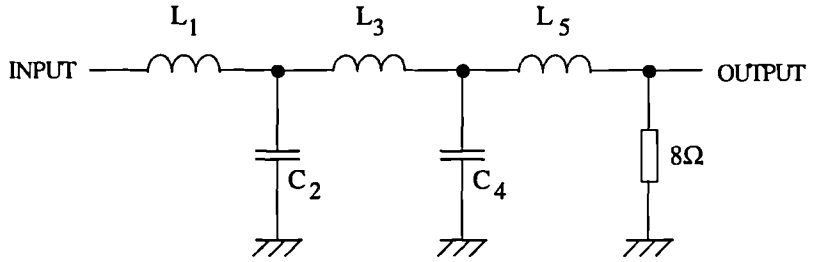


Figure 7.4.4.a : Design Two, Ideal Component Schematic.

With design tables the component values and from these, their parasitics, can be estimated :

		ø = 0.45 mm, 1.0 mm	
L_1 = 37.83 µH			
C_2 = 648.2 nF	N_1 = 137 turns	R_1 = 0.69 Ω	0.14 Ω
L_3 = 33.84 µH	N_3 = 129 turns	R_3 = 0.65 Ω	0.13 Ω
C_4 = 342.2 nF	N_5 = 61 turns	R_5 = 0.31 Ω	0.062 Ω
L_5 = 7.566 µH			

Figure 7.4.4.b : Component Values for a 52 kHz,8Ω Load Butterworth Filter & Estimated Resistances

With this filter the carrier second harmonic is suppressed by 110 dB and the transient switching currents (near the counter clock frequency) are reduced to below noise or radiated levels. The load impedance to the switches is approximately 38 µH. The inductor turns and impedances are calculated from first principles for winding the inductors on 5 cm. diameter toroidal cores, 1.5 cm. thick, with μ_r = 1.0 (ie. air cored to avoid saturation).

Knowing that the capacitors would have to be replaced by a value from a discrete set, C_2 was modified to 680 nF and C_4 was modified to 330 nF. After nodal analysis with the real component values, the frequency and phase responses, and the input impedance, are as shown below :

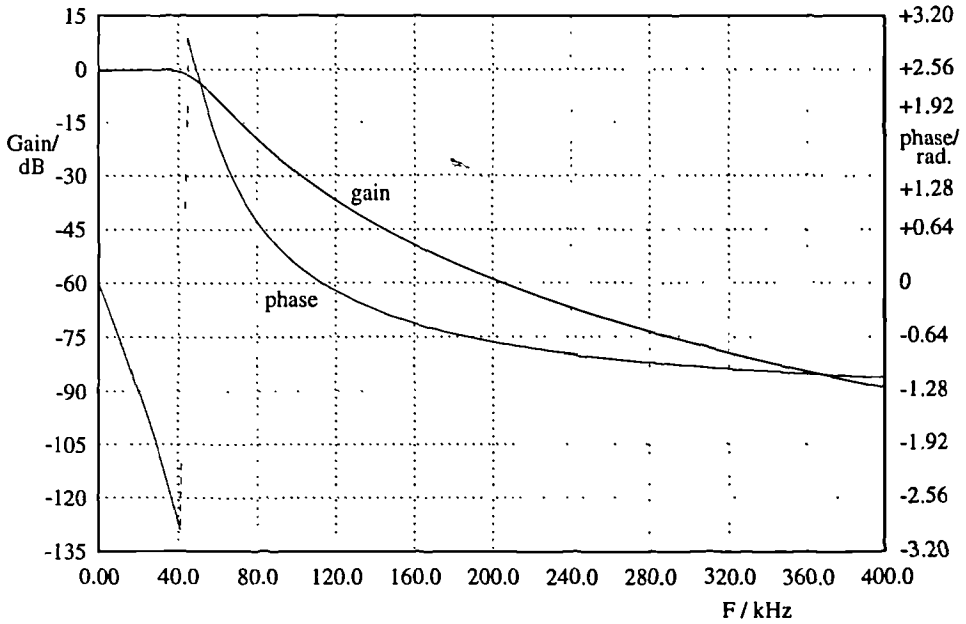


Figure 7.4.4.c : Design Two, Simulated (Real Component Values) Magnitude & Phase Responses.

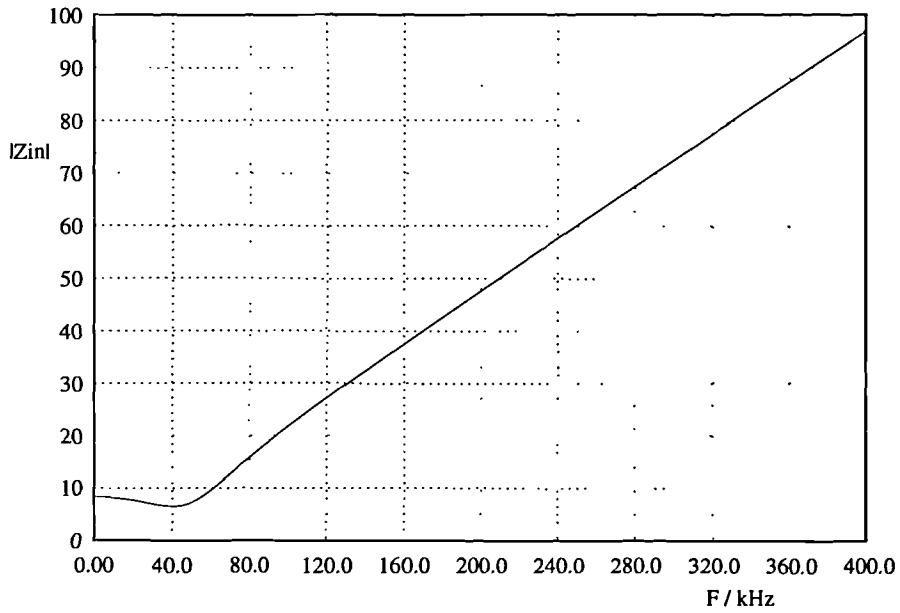


Figure 7.4.4.d : Design Two, Simulated Input Impedance as a Function of Frequency.

From these spectra this solution can be seen to meet the design specifications as discussed in section 7.4.1. It is worth noting that this filter yields slightly more suppression in practise (probably as a result of C_2 being slightly higher in value than required, although still within tolerance). Also, noise power in the load may well be large as a result of the filter input impedance falling near 40 kHz and noise shaper NTFs that rise sharply after 20 kHz. In practise, heating or linearity reduction from this could not be detected although tests at high power levels might reveal problems arising from this.

7.4.5 : Design Three Solutions

Design objective three requires the simplest of the output filters attempted, being only a second order filter (as shown in figure 7.2.1.a). Measurements of an inductor, using a 3" long, ceramic core allowed 21.3 μH to be achieved for an 85 kHz (- 3 dB point) Butterworth filter using a 165 nF capacitor. The inductor 's series resistance was only 0.1Ω so the actual and ideal performances are very similar. This filter has negligible passband rolloff, has very near linear phase in the passband, and suppresses the first carrier by 25 dB. Performance curves are shown below :

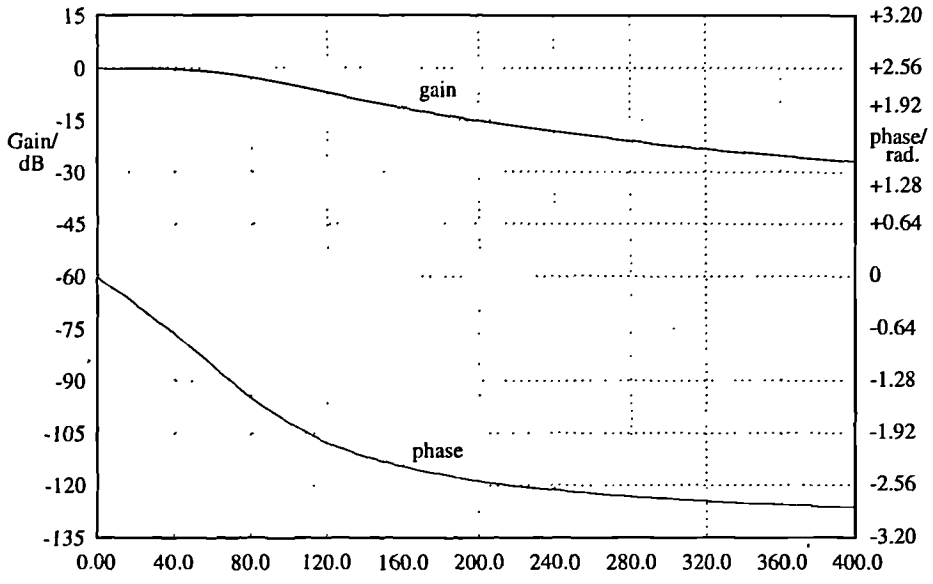


Figure 7.4.5.a : Design Three, Simulated (Real Component Values) Magnitude & Phase Responses.

In a circuit modification to achieve more carrier suppression, the capacitance was changed to 385 nF which makes the cutoff slightly more resonant. This increases the passband ripple to 0.6 dB and increases the carrier suppression to 35 dB as shown below :

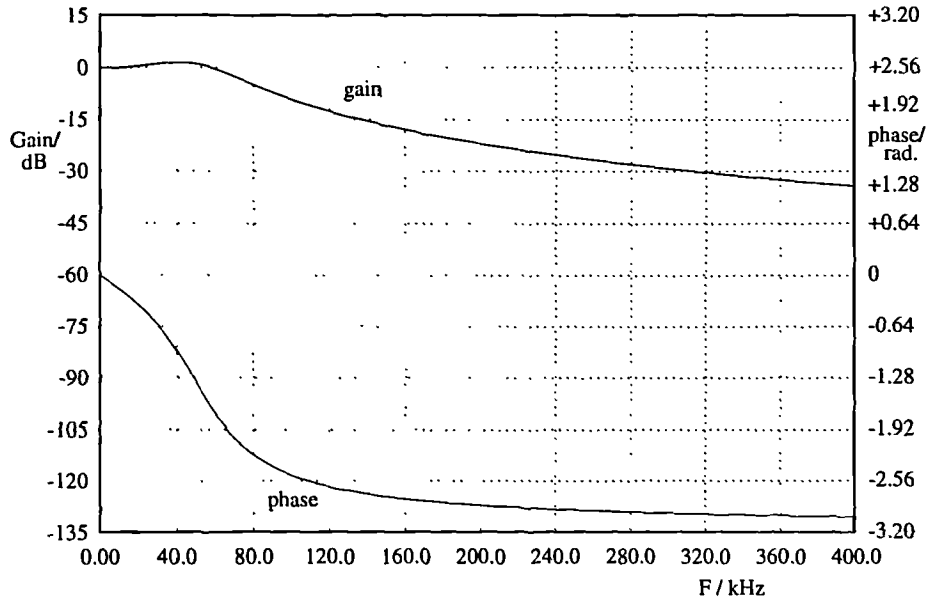


Figure 7.4.5.b : Alternative Solution to Design Three, Simulated Magnitude & Phase Responses.

This change makes the filter appear more capacitive to the power switch, lowering the input impedance of the filter near its cut-off frequency. Since this is likely to coincide with the beginning of the passband of the NTF used in the DSP stages' noise shaping, more current is wasted on amplifying noise with this design change. This leads to a reduction in the efficiency improvement achieved by the additional carrier suppression than expected, but is still very worthwhile. A graph of the input impedances of the two designs is shown below.

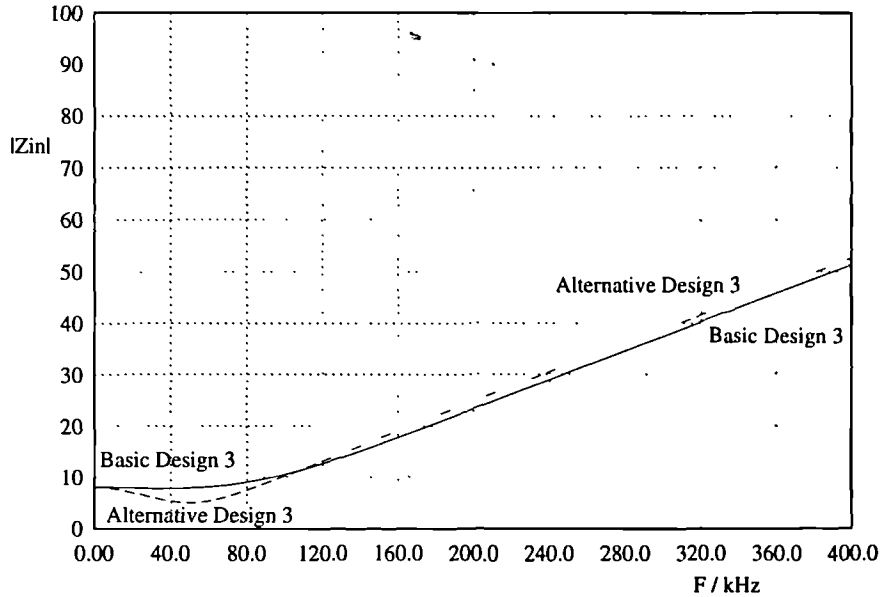


Figure 7.4.5.c : Design III & IIIa , Simulated Input Impedances as Functions of Frequency.

7.4.6 : Star Filter Configuration

By taking the simple second order filter of design three and considering its input node as the branch of two MOSFET drain connections and the filter input, the design can be converted from a 'T' at the input to a 'Δ'. Observing that the drain resistances are small, and that the switches should operate exactly out of phase, the equivalent circuit can be transformed by simply doubling the inductors :

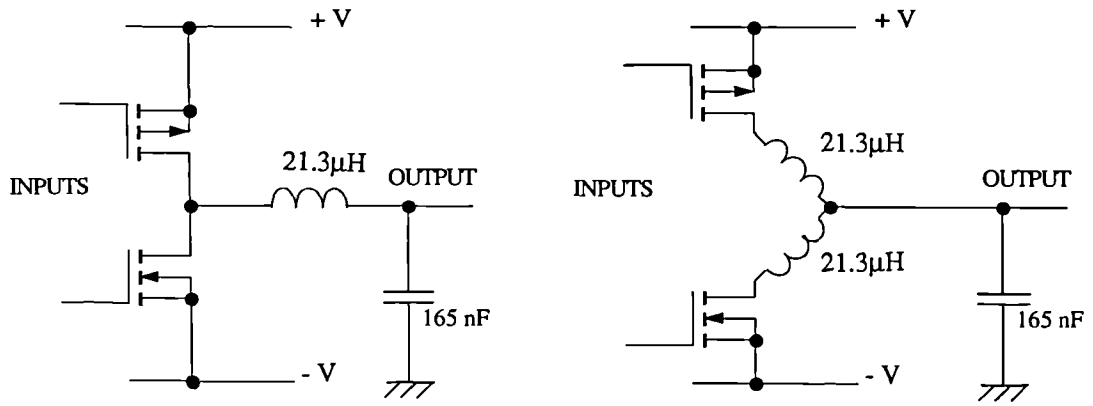


Figure 7.4.6.a : Conversion to produce the Star Filter Configuration.

It should be noted that it is not the filter which has been converted from star to delta, only its input node, and that this is only valid if the layout impedances are small (which they should be for useful output current to flow).

With this revised configuration, the shoot through currents caused by overlap in the conduction cycle of the two MOSFETs can be almost eliminated since they now have to pass through two inductors (rather than none). Since the overlap duration is small, the current change is limited by the inductors and both MOSFET heating and output errors associated with shoot-through can be avoided. Furthermore, by making these inductors slightly variable, the asymmetries in the pulse edge shapes can be minimised, allowing lower THD figures from the switch as a whole. For a full bridge implementation, four such inductors would be required so space considerations may limit the usefulness of this approach in a commercial design.

7.5 : Summary.

Output switch configurations for PWM DACs are mainly restricted to class D type as a result of using high modulation depth and the need for power-efficient switching action. The fundamental components are a pair of power transistors switching in a complementary fashion to connect a high or a low voltage to the load. By switching over a larger voltage range, larger load power can be controlled, hence performing amplification as well as D/A conversion. By allowing the user to set the power supply voltage used, a form of 'volume' control can also be incorporated. High speed performance is particularly necessary since oversampling is commonly used and switching rates between 200 kHz and 1 MHz are expected.

Edge accuracy in shape and timing are of particular concern because the accuracy of the signal remains as high as that of the input despite noise shaping in the earlier stages of a PWM DAC to reduce wordlength. Since signal amplitude and phase information are both carried in the timing of pulses in PWM, errors in the edges translate to pulse area and pulse position changes which can be related to better understood parameters (such as SNR) commonly used with conventional A/D converters. For 16 bit data sampled at 44.1 kHz (as stored on a CD) edge timing accuracy should be within 115 ps, RMS (assumed Normally distributed). This can be achieved by advanced devices at sufficient power levels.

Pulse amplitude error is a problem that introduces both noise and distortion which cannot necessarily be removed at a later stage. Signal dependent errors such as arise from capacitive coupling and supply droop tend to introduce harmonic distortion. For this reason power supply regulation is essential, a switched mode type being particularly suitable for its high efficiency. If used, a switched mode regulator should be locked to a fraction of the pulse repetition rate to avoid intermodulation with the output PRR. It should be noted that even if linear regulation is used, the overall power supply requirements are smaller than those for a class A, B or A/B amplifier. Optical circuits can be used to allow level shifting without capacitive coupling within drive circuitry but this introduces unacceptable jitter unless a additional circuit complexity is used to carry out resynchronisation of the pulses.

Crossover distortion can be introduced by using anti-parallel diodes or MOSFET switches with a high channel resistance. The problems associated with the anti-parallel diodes can be eliminated by appropriate offset voltages being used for recirculating currents. Output device resistance can only be reduced by using better design geometries or materials.

Two types of output switch can be used, a half-bridge or full-bridge, the first giving better accuracy through its simplicity although more susceptible to power supply variation, the second giving four times the power handling capability. Both should be used with output filtering of which several designs have been discussed to suit output power requirements and output switching component suppression requirements. One special filtering structure is recommended for its ability to eliminate high switching currents that otherwise cause signal distortion and output device heating and failure.

Designs switching at hundreds of kilohertz but at low power levels have been constructed and suggest that higher power designs are perfectly possible. Theory suggests that designs capable of handling up to several hundred watts can be achieved with current technology resulting in THD figures of less than 0.003% and SNR figures of at least 90 dB.

Chapter 8 : Summary and Discussion

8.1 : Summary of Work.

8.1.1 : In General

From the outset, this thesis, and the work supporting it, has been aimed at solving one central problem : how can high-power, analogue signals be generated from digital data ? Solving this is the key to implementing a 'Digital Amplifier' which can produce high quality signals (with linearity and resolution comparable to that encoded on a compact disc) at high power levels such as required for driving a loudspeaker.

At the core of this problem is the conversion process between digital data and analogue signal. Digital, pulse-width modulation (DPWM) was chosen to do this, because of its potential for high accuracy and high efficiency resulting from minimal switching transitions per second. In isolation, this process is neither feasible with high resolution, nor linear, time-independent (LTI), so additional digital signal processing (DSP) and high speed, high accuracy circuits have been designed to overcome the shortfalls.

A block diagram including the essential elements is repeated below from for reference :

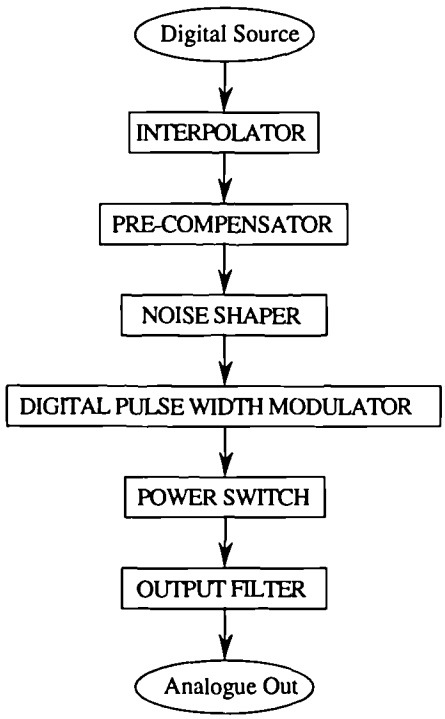


Figure 8.1.1.a : Basic Block Diagram of a System Proposed for a Digital Amplifier.

To overcome the resolution and linearity restrictions of DPWMs, additional system elements have been proposed and then investigated using theory, computer simulation and measurement of prototypes. These elements have been shown to introduce further restrictions on the system (eg. noise shaping requires oversampled data) and in some cases, further limitations arising from secondary interactions between each other (eg. between the DPWM and the noise shaper).

When investigation started into producing a digital amplifier, five problems had already been anticipated, namely :

i) Efficiency limitations in the output stages arising from output pulse repetition rates (PRR) of 700 kHz or more, as required for oversampled data streams.

ii) Resolution limitations arising from the use of 1st or 2nd order, sinusoidal noise-shaping [RIB91] which only enables limited wordlength reduction unless large oversampling is used (typically SNRs of less than 70 dB are the best that could be achieved with sample rates below 1 MHz).

iii) Linearity limitations in the conversion stage arising from the use of trailing-edged modulation (TEPWM). This prompted the investigation of alternatives such as two sided, or pre-compensated DPWMs.

iv) Excessive counting rates within the DPWM module arising from the combination of a fine quantisation of pulse widths, required for a high resolution input, and the relatively large bandwidth of the signal that has to be represented. These becomes even more impractical when, as previous work suggested [SAN83], oversampling is used to reduce the harmonic distortion arising from uniformly sampled PWM (ie. suggesting provision for yet larger bandwidths).

v) Construction limitations, restricting the amount of processing that can be achieved in the available sampling periods, the accuracy to which the circuitry can reproduce the data, and the edge resolution in the DPWM (via the maximum counting rate) and hence the maximum signal resolution.

Assessment of analogue PWM based amplifiers showed that output switches required to operate at high PRRs are particularly difficult to construct. Furthermore, the amount of additional signal resolution or linearity that could be achieved by high PRR was found to be small. This forced an emphasis on the DSP aspects of the system, and so a compromise between five factors was looked at to solve the problem. These variables are :

1) the oversampling ratio, which in most cases will determine the PRR,

2) the noise shaper order, with higher orders allowing higher resolution in smaller wordlength,

3) the wordlength applied to the DPWM, which determines the counting speed required,

4) the modulation type, essentially a choice between symmetric and trailing edged PWM,

5) the modulation depth, with smaller values giving lower distortion and efficiency,

An initial trial system using sinusoidal noise shaping quickly showed a balance is required between the order of noise shaper and the oversampling ratio used. For example, in chapter 6 a fourth order noise shaper was shown to give best output SNR at eight times oversampling. At lower orders the reduction of wordlength before the DPWM was small so that the resulting counting rates was not feasible. At higher orders, noise sidebands were found to reduce the output resolution so that no further gains could be made. This compromise does limit the SNR, which is at best 88 dB after modulation, but the noise is predominantly near 20 kHz where the ear is insensitive (after E-weighting this becomes 92 dB, only a 6 dB degradation compared to CD quality). With a counter speed of 100 MHz, an 8 bit DPWM was constructed and a near-symmetric modulation (AOAPWM) was attempted with 88% modulation depth. However, high frequency input signals still caused concern since total harmonic distortion (THD) was as high as 0.16 % (for a full-scale, 10 kHz input tone).

By studying the spectral performance of analogue variants of pulse width modulation and extracting the best aspects of one, a pre-compensation technique was found which can reduce the distortion produced by the DPWM (see section 5.2.6). Naturally sampled PWM exhibits no signal harmonic distortion and no intermodulation between tones in twin-tone input signals, so an algorithm mimicking the action of an analogue natural sampler [HIO89] was used to find the pulse widths that are required for zero harmonic distortion. This work has been taken further [GOL92, SAN91b, SAN92] and recognised in an alternative form [CHE92] and is now well understood as 'Pseudo-Naturally Sampled' PWM. This kind of pre-compensation cannot improve on the performance of the modulation it imitates, naturally sampled PWM; interactions between the noise shaper and the DPWM are not eradicated (such as re-modulation of shaped noise into the baseband of the output spectrum). These effects have been found to be the theoretical limiting factor in the performance of open loop, PWM based DACs.

Without harmonic distortion from the modulation type, the need to use oversampling to reduce harmonic distortion and the requirement for symmetric modulation types can be removed. This allows the oversampling factor to be reduced to the point at which sidebands of the PWM input signal reappear in the baseband of the output spectrum at higher levels than the inherent quantisation noise (see section 2.4.5). For 16 bit systems this is at about 9.6 times the signal bandwidth, so 5 times Nyquist sampling is suggested as a convenient integeric ratio.

To reduce interactions between the noise shaper and the DPWM, the spectral shape of noise produced by the noise shaper has to be either tightly controlled, or known and compensated-for in advance. Both these approaches have been investigated in chapter 4, with a view to higher resolution systems and allowing simpler DSP in the existing systems for a given performance. Compensating for the noise shaper's noise contribution at high frequency is very difficult since estimating it in advance is not feasible as a result of the non-linearity present in the quantisation process. Measuring the error signal and compensating for it is also difficult since this is within a feedback loop and would have to be done almost instantaneously to correct for high frequency components. Compensation for the noise is also marginally detrimental to system performance, because sidebands of the noise arising from mimicking natural sampling lead to higher baseband noise levels than if the noise is modulated as uniformly sampled. Thus extensive work has been put into designing shapers so that the spectral shape of its output noise causes minimal re-modulation problems after pulse width modulation.

The spectrum of noise from noise shapers can be estimated from the noise transfer function (NTF) and assuming requantisation noise to be additive and white. As shown in chapter 4, substantial SNR improvements have been made by optimising NTFs for low power gain, and minimal use of frequency bands to which the PWM is known to be sensitive,. In the same way, interaction between particular types of DPWM and the noise shaper for can be reduced (notably 2SCPWM and DSPWM by treating rising and falling edge components separately - as shown in section 4.5.5).

With improved systems as a result DSP which greatly reduces noise and distortion, small circuit effects become the largest remaining problems to solve (ie. beyond considerations of maximum counting speed and using two level output). These are largely centred in two areas : getting sufficiently accurate edge timing and avoiding pulse amplitude errors (see section 7.1); both of these are significantly more difficult to solve at high output power levels.

Accurate edge timing can be achieved by latching the DPWM signal with a high purity clock source. Ideally this clock source would be a local oscillator, and crystal based, but this introduces the problem of synchronisation with the data source. In some instances the DPWM clock can be divided down to provide a synchronous clock for the data source, but if this technique cannot be used, an asynchronous sample rate conversion will be required at the input (for example, if different data sources and sample rates are required). With this technique, near-unity interpolation or decimation ratios can also be achieved, permitting fine-tuning of the oversampling ratio for the PWM, independently of the input sampling rate. This also enables standard interpolator chips to be used for non-standard oversampling ratios; for example 10x oversampling can be achieved by 1.25x asynchronous sample rate conversion followed by (standard) 8x synchronous sample rate conversion (interpolation). With these advances, the noise shaper and DPWM parameters become more flexible, enabling designs to achieve at least CD quality output for the D/A conversion part of the amplifier as demonstrated in section 6.4.8.

Switching accuracy in the output is now the limiting factor in the digital amplifier topology as presented in figure 8.1.1.a . Circuits have been found for ensuring optimum operation of the output switch, centred on four techniques :

a) The output topology should be a full bridge type to remove as much of the switching component of the load current away from the power supply regulator. This should also be supported by significant decoupling near the bridge itself to provide recirculating current paths.

b) The output PRR should be reduced as much as possible to simplify the switch. The minimum should just prevent modulated noise sidebands from reducing SNR in the baseband. This can be achieved with the fine tuning of the oversampling ratio as mentioned above, and by using double sided modulation types.

c) The power rails should be as stable as possible since the output switch operates as a voltage source. This requires regulation, and is best met by switched mode regulation which is consistent with the high efficiency operation of the output itself.

d) The output filter should be inductive so that recirculating currents are kept small, especially at the carrier frequency and multiples of it. Careful integration of the filter and the output switch can also be used (see section 7.4.6) to relax circuit timing limitations such as the need to avoid simultaneous conduction of both devices in each half bridge section.

In short, signal processing and circuitry have been presented which can be used to form a digital amplifier. Most implementation problems have been resolved to produce CD quality output despite interactions between blocks required for solving each of the implementation problems. This has been done using analysis of theoretical performance from analogue PWMs, to make inferences that would apply to digital PWMs. Computer simulation of the system performance has been used to confirm the nature of complex interactions within these systems and to investigate the sensitivity of the overall performance to non-ideal circuitry. With this information, prototypes have been constructed, and measurement of them has validated the original analysis and the concept of a digital amplifier, as well as providing more information about the relative significance of implementation difficulties.

8.1.2 : Additional Points About Noise shapers

It is worth noting that in this work, new noise shaper design methods have been developed, both in terms of objectives and techniques. Conventionally, noise shapers use sinusoidal transfer functions which have been said to contribute *minimal* noise power to the baseband [TEW78]. By designing for *tolerable* noise levels in the baseband, the noise power gain has been reduced dramatically. This becomes important in almost all applications of noise shaping although it has been used to advantage in this work to reduce interaction between the noise generated in the requantisation process within the noise shaper and the PWM process which follows it.

A second feature of work on noise shapers within this thesis is the hypothesis that target functions should be defined for minimum loop delay in the noise shaper rather than minimum power gain. This builds on existing work suggesting that minimum power gain can be achieved for a given shape of NTF by ensuring it has minimum phase-lag properties [GER89].

A third aspect of noise shaping that has been developed in this work is an application of feedforward error correction as well as error feedback. Conventional noise shapers only make use of error feedback systems, placing correction quanta close to the ideal point in time but always later. Error feedforward and approximate error feedforward techniques suggested here, permit correction quanta to be placed more symmetrically about the ideal point in time, reducing the second moment of the error and hence more closely following the input signal.

8.1.3 : Additional Points About Fast Digital PWM Design

Circuitry for digital PWMs has been developed simultaneously with this work and demonstrated with a series of prototypes. As a result of these, circuit design techniques based on critical timing-path analysis have been developed allowing simultaneous synchronous and asynchronous control to be applied. From the outset, designs were attempted in 'commercially available' logic families to allow reliable prototypes and demonstrate realistic cost implementations. The most advanced family used is the 74AC CMOS logic family which is intended for system speeds up to 60 MHz. Through the advanced design techniques employed, system speeds well above this were used successfully (101, 126, 144 & 180 MHz).

Several design approaches have been found essential to successful implementation of fast DPWMs. Firstly, pulse edge turn-on and turn-off times are best controlled by the time at which a counter passes zero (not by comparison to the required data value). This permits the same end-condition to be used for all data values, simplifying one of the fastest parts of the circuit, and allowing end time anticipation since penultimate values are known. Secondly, load periods for counters should be kept separate from counting periods so that fast start-up of counters can be used. This fits in well with output circuitry which require guard bands - a convenient moment to allow presetting of counters. Thirdly, capacitive load on intermediate control signals should be inversely linked to the speed of the signal; capacitive load (ie 'fanout') should be minimised of for high speed signals, and can be heavier where this matters less. This has been found to be consistent with asynchronous counting topologies where clock signals are numerous, but each is little used.

8.1.4 : Additional Points About Open-Loop Linearisation

A mentioned in section 8.1.1, numerous pre-compensation techniques now exist for linearising the action of digital PWMs. These fall into three groups - firstly, techniques which manipulate the pulse so that it emulates another modulation type. For example, delaying a pulse by half its width converts trailing edged modulation into symmetric modulation, dramatically reducing the harmonic content of the output.

Secondly, there are a complete family of pre-compensation techniques based on the concept of copying an analogue PWM (performed by a comparator) [HIO89]. These range from simple linear interpolation [CHE92], to more sophisticated polynomial interpolation (pseudo-natural PWM) [GOL92]. On this idea, there are further techniques which have been developed by other authors such as using a mixture of sample rate increase and linear interpolation to reduce processing complexity [PED94]. Two of particular note use a balance of pre-compensated and uncompensated data to provide either a balance between harmonic and sideband distortion [MEL91] or a reduction in odd order harmonic distortion [PAU92A].

Thirdly, there is one published technique for the frequency domain minimisation of the in-band distortion components normally associated with digital PWMs [HAW92]. This takes the sophisticated approach of using an adaptive filtering technique to choose appropriate compensation dependent on the input data set. Although not developed to CD quality yet, it is felt that this approach should be better than that of copying natural sampling (the analogue PWM variant) since it aims at eliminating all distortion in the signal band, not just the harmonic content. It is interesting to note that this approach uses a time-variant, linear technique which might be the expected antidote to a time-variant, linear system, rather than the natural sampling emulation which employs a time-invariant, non-linear technique. Both techniques are 'non-LTI' but it is thought that cancelling one error with another is less likely to cure PWM's distortion more effectively than correcting for the error in the first place, and it is more likely to be sensitive to implementation non-idealities.

8.1.5 : Additional Points About Closed-Loop Linearisation

Knowing that output power stages are the quality-limiting part of the digital amplifier, suggests that some correction for the output which encompasses switch problems should be used. With this in mind, feedback from the output of a DPWM has been analysed, along with various alternative structures to make this idea more practical. Several classes of error have been identified and can be treated collectively of separately. These are the quantisation error required to modulate a practical wordlength within the DPWM (previously accommodated by noise shaping), the harmonic and sideband error from the DPWM itself (previously accommodated for by pre-compensation) and the switching errors themselves (arising from non-idealities in practical switching topologies).

Difficulties arise in compensating for the errors at the output of the DPWM because they are usually small as a result of preprocessing (<0.1%) and in the presence of a large signal; as a result, large errors are incurred in finding the necessary correction signals. This is compounded by the sample rate of the output being very high so that evaluating the correction signal requires significant low pass

filtering, either to allow 'digital evaluation' of the correction signal (ie. A/D conversion of the output followed by decimation and comparison with the input) or 'analogue evaluation' of the correction signal (analogue low pass filtering of the output followed by A/D conversion and digital comparison with the input). Both routes also involve A/D conversion which may well inject more errors than the PWM DAC itself.

If the output switch is left open loop an attempt can be made to linearise the DPWM itself by applying local feedback around the DPWM only or the DPWM and noise shaper together. These are both very difficult because of the decimation of the output which is required. Delays incurred to discriminate between signal content to compensate for, and aliases to remove in decimation, render such a system useless without very high oversampling (> 200x for 16 bit resolution), contradicting the original objectives of amplification (which is limited by output switch pulse repetition rate).

One alternative which has been found useful for signal-level D/A conversion is to emulate the distortion that would be expected from the DPWM by using precalculated error tables for groups of input data sets [CRA92]. These stored correction signals can be fed back at the input sample rate, avoiding decimation filtering, but because the errors have both recursive and forward-in-time consequences, only limited correction can be applied. High oversampling (64x) is again used to reduce the PWM's non-LTI effects so that high linearity and low noise gain can be achieved. Like the adaptive filtering used in [HAW92], the tables contain functions representing time-variant, linear mappings, supporting this as a good approach for correction of PWM's non-LTI behaviour. Published results indicate very high linearity can be achieved in this way (approaching 22 bit dynamic range if simulation results can be maintained by practical circuitry).

8.2 : Discussion of the Limits of PWM DACs.

8.2.1 : Resolution

Three factors are currently seen as the limitations to the resolution that can be achieved from PWM based DACs. These are the interaction of noise shaping with digital PWMs, the dynamic range of circuitry used and the time jitter of whatever clock sources are used. All of these sources of error are significantly smaller than the problems known to hamper analogue PWM based DACs and in some cases, smaller than problems that exist in power amplifiers and DACs based on alternative conversion processes.

The characteristics of the noise shaper / DPWM interaction are low level output noise which rises with frequency as shown in chapter 2. This is problematic for high amplitude signals only and usually when high amplitude and high frequency input are combined. The low-level, high-frequency nature of the problem is of less concern than other DAC defects (such as non-monotonicity or missing codes) since psychoacoustic measurements suggest that this noise is barely audible (see section 1.5). Furthermore, the noise is more severe when high amplitude inputs are applied, which are likely to mask such defects. For these reasons PWM DACs are nearer in quality to conventional DACs than spectral measurements such as SNR over the 0-20 kHz band would imply.

Output circuitry dynamic range has been highlighted as another problem which PWM DACs suffer from because the output quality is dependent on the power supply being a true voltage source. This problem already exists in conventional class A, B and A/B designs (although to a lesser degree). At least CD quality output at powers up to 100W should be possible if appropriate development of output supply regulation in conjunction with inductive filtering is used. Like the noise-shaper / DPWM interaction, this problem becomes most acute for high amplitude signals and hence does not have such severe consequences in reducing the output quality as measurements would imply; masking effects are likely to ameliorate this effect. Using a voltage source to drive a current transducer, such as loudspeaker, is less than ideal unless impedance variation is taken into account (their input impedance often varies as a function of frequency and amplitude). Impedance matching, or frequency response modifications may be necessary in a commercial design, but since PWM DACs are most suited to 'active speaker' applications where the load characteristic is known, this need not be a significant problem.

Clock sources can be chosen so that pulse timing accuracy is sufficient for CD quality although they directly influence the output quality from any pulse stream DAC. Regenerated clocks from input data streams have been identified as poor sources for the accuracy required in PWM D/A conversion, but local clock sources have been shown to be adequate in section 7.1. Crystal based clocking is a particularly good timing reference but if used with a remote clock controlling input data rate, asynchronous sample rate conversion (see chapter 3) or considerable amounts of interface FIFO memory should be used to avoid dropping or missing data. Asynchronous sample rate conversion does introduce time-jitter into the signal, but this has been shown to have very small consequences, and is far preferable to having sampling instant jitter than edge jitter (which converts directly into amplitude noise in PWM systems).

8.2.2 : Linearity

Linearity in PWM based DACs is largely dependent on the modulation type used and also affected by how ideal the operation of the output switch is. Output switch linearity (including output filter linearity) is difficult to compensate for, and detailed analysis will be left as an area for future research. Fortunately most non-idealities in the switch are related either to edge shape modifications (typically low pass filtering) or load current dependencies (which can be estimated from the input signal data values). In a system where low pass filtering is already employed and dominant load currents are in the lower 10 % of signal frequencies, significant improvements can be made by global feedback providing adequate linearity A/D conversion is employed in the feedback path. Load current dependencies are usually low order non-linearities and may be approximately predicted to reduce the feedback loop gain that would otherwise be required.

Non-linearity arising from the choice of modulation type and pre-compensation technique has already been discussed (sections 8.1.4 & 5 and chapters 2 & 5). In the conversion stages of a digital amplifier at least 16 bit quality has been achieved with pulse repetition rates at which output power switches can operate (<500 kHz) and higher resolution can be achieved if psychoacoustic (E-weighted) assessment is considered adequate. If only signal level conversion is required, better linearity (22 bit) has been demonstrated by using feedback with error-emulation and higher oversampling [64x, CRA92].

8.2.3 : Efficiency

PWM based digital amplification really stands out from other techniques when efficiency is considered. Class D output stages are particularly efficient with figures of up to 98% being commonplace in power supply regulation applications. PWM based DACs employ counting at high frequency so some efficiency is lost before the power stage of a digital amplifier in the generation of appropriately timed pulses, but since asynchronous counting is the preferred counter topology for speed, few counters actually run all the time. When compared at signal level, PWM based D/A conversion use more power than current-summing or resistor-ladder DACs (typical figures : 2W for PWM-based, 600 mW for current-summing, 250 mW for a resistor ladder DAC). At signal levels this power is seldom of concern unless it is used in portable equipment; high integration versions of a PWM based DAC would reduce this power consumption by a factor of 10 or more. Lower quality may be tolerable in such applications (eg. personal stereo) allowing the counter frequency and hence current consumption to be dropped by a factor of 4 or more.

At high power (>50W) the increased efficiency of class D outputs is very useful, permitting use of smaller heat-sinks and smaller power supplies, and allowing a greater power concentration per unit volume. This would allow lighter, cheaper, more compact amplifiers to be used than are currently available. In the limit, with both the power supply regulator and the output stage operating at, say 95% efficiency, an overall output could be at least 90% power-efficient (which compares with at best between 50 % efficiency for class A and 66 % efficiency for class B designs).

8.3 : Suggestions for Further Work

8.3.1 : Better Pre-compensators

PWM's inherent non-LTI behaviour remains a stumbling block for high resolution D/A conversion (and in the limit this will apply to the digital amplifier too). At present, CD quality output has been achieved (16 bit, 0-20 kHz) but future amplifiers may well require higher resolution for digital audio volume implementation or higher quality signal rendering. Pre-compensation is to date flawed by attempting to copy natural sampling (which has its own limits) or flawed by stability limitations within feedback networks which force the use of high oversampling to ensure that adequate signal bandwidth can be linearised. Three techniques are expected to help in this area, which in decreasing order of complexity and anticipated effectiveness, are :

Since PWM has been identified as a linear, time variant transformation of signal data to pulse widths, this author believes that time-variant, linear filtering (TVLF) offers the best route to compensating for PWM's non-LTI behaviour, although it is recognised that the complexity of such a scheme will be large. Non-linear system identification (Weiner or Volterra) may assist with producing TVLF but frequency domain moment matching as already published [HAW92] is recognised as the most successful development in this area to date.

Interaction between noise shaping and a pre-compensated DPWM has been reduced in this work by reducing the stimulus to the system (the high frequency noise - see chapter 4) rather than the system's non-ideality itself (the sideband behaviour - see chapter 2). Simpler strategies than TVLF may exist, such as sub-band coding (spectral masking of shaped noise to within sub-bands of the audio band - eg. MPEG, layer II). It is not yet known whether the levels of intermodulation of such a technique with the DPWM would make the sub-band coding inappropriate and hence a revised coding strategy should be adopted. Such an approach is appropriate for a final system stage where information loss can be tailored to the application and should allow the counting rate within the DPWM unit to be reduced considerably.

Since interaction between noise shaping and a pre-compensated DPWM has been made small by steps already taken (specially shaped noise transfer functions within the noise shaper itself), correction for this may only need to be slight. This author believes that if appropriate pre-compensation is followed by mild linearisation of the remaining errors from the DPWM by feedback (using emulation as described in [CRA92] but for sideband error alone), the low oversampling DAC or digital amplifier could be linearised further with little effort.

Similarly, the optimisation of appropriate NTFs could be improved by using system optimisation in place of just noise shaper optimisation. System interactions could be minimised within the baseband by using emulation of the output from a PWM in place of output signal decimation for analysis. This would speed up the optimisation of the system so that noise shaper / DPWM interactions could be minimised systematically rather than heuristically.

Despite the comments above relating to PWM at the core of a digital amplifier, examples as put forward within this thesis do provide adequate quality for state of the art amplifiers, as has been substantiated by academic [HIO92], patenting [SAN91b] and commercial publications [FAR93].

8.3.2 : Pulse Position, Pulse Width Hybrid

PWM has been proposed at the core of a digital amplifier because of its low edge rate compared to bitstream or other D/A systems employing single-bit output (and hence its high efficiency and potential for minimal errors). Other systems employing similar numbers of edges per second could be considered on the same basis.

Existing PWM outputs utilise width and position definitions that are singular (ie. they both describe the same signal information). For example, symmetric PWM has a delay from the start of each pulse period to the start of the pulse which is one half of the time in the pulse period which is not used by the pulse width. If the localised position and the width of the pulse are considered to be independent variables, more degrees of freedom are released for manipulating the information carried by the pulse. A bi-variate table containing paired compensation information could then be generated, attempting to make best use of this additional degree of freedom, and hopefully achieving better linearity as a result. If such a table is used as a look-up between the noise shaper and the DPWM it can linearise the DPWM, eliminating interaction between the two as well as the distortion usually associated with simpler modulation schemes.

The look-up table size for a linearised PPM-PWM hybrid is likely to be large since the errors in simple modulators are known to span several samples, implying that several samples need to be considered when determining the appropriate position and width pair, in a sequence of such. Systematic functions are expected to describe these so a compromise between the table size and the mathematical complexity of a function evaluation is anticipated. If such a system is incorporated within a feedback loop, stability constraints are expected to steer this towards the faster (look-up table) approach, even if this produces a poorer approximation to the ideal output pair.

8.3.3 : Better Output Stages

Difficulties are anticipated in producing an output stage which does not produce significant EMI. To date, the severity of this problem has not been quantified but there exists a potential problem in this area. Solutions vary from repeated shielding (of both electrical and magnetic fields) to switching topologies that can support resonant edges. Lower PRRs help to reduce this problem although low pass filtering of the carrier becomes increasingly difficult without compromising the signal band. Potential solutions to this lie in pre-boosting the high frequencies of the signal band to allow for in-band roll-off, or notch filtering of the carrier fundamental to avoid using high order filters.

Placing the output filter within the same screened enclosure as the output switch prevents EMI apart from that which leaks unintentionally through the screening. If the filtering of the output is inadequate to permit adjacent radio and television receivers from proper operation, digital amplifiers may be limited to low power (eg. portable) or active speaker systems. To determine this, higher power output stages than used in this research need to be constructed.

Whether or not class D is used for a full audio bandwidth amplifier, it should be used for at least low frequencies where its linearity and resolution is good, and its efficiency excellent; if necessary, this could be supplanted by a low power class A, A/B or B amplifier for higher frequency bands.

Appendix 1 : Open & closed loop PWM DAC simulation software.

A.1.1 : Signal generators

Three programmes are listed below, 'SIGGEN.PAS', 'NATSIGEN.PAS' and 'DITHER.PAS'. These are used to generate signals with known properties with which to test architectures and algorithms relating to PWM DACs.

'SIGGEN.PAS' is configured to generate single tones with up to 16 bit accuracy, and store the result in compact (integer) format. 'NATSIGEN.PAS' generates two data streams from the same signal with no phase offset and up to 24 bit accuracy. The two data streams are stored '.wav' format as used by Audio Precision's "System One/Dual Domain" test set. The left and right channels are uniformly sampled and naturally sampled versions of the same signal and can be used for developing open loop pre-compensation algorithms to emulate natural sampling. 'DITHER.PAS' also uses '.wav' format output, producing two dither streams suitable for use with any requantisation stage. Compilation options can be used to generate dither suitable for either node A or B dither, with rectangular PDF or triangular PDF, high-pass characteristics or self cancelling properties for use in dual quantiser shapers.

SIGGEN.PAS

```
PROGRAM unif rmly sampled 16 bit integer_sinewave_generator ;    {$N+,E+}
          { ! amended t  include dither }
TYPE integer data buffer = ARRAY [0..16383] OF integer;

CONST sampling frequency = 352800   ; { 8x oversampling  }
      signal frequency   = 1 01     ; { Fc Fv irrational }
      no  f bits         = 16       ; { variable limit ! }
      no of samples      = 131840   ; { pascal integers. }

VAR file out                    : file                                  ;
    unif rm_data                : ARRAY [0..8] OF ^integer_data_buffer ;
    temp int                    : integer                               ;
    c unter,  ffset             : l ngint                               ;
    percent d ne, index         : byte                                  ;
    amplitude, frequency, data  : single                                ;

BEGIN
   IF PARAMCOUNT <> 1 THEN
   BEGIN
      WRITELN(' utput amplitude required (*db.int* extension added).') ;
      HALT                                                              ;
   END                                                                  ;
   IF MEMAVAIL < 294912 THEN
   BEGIN
      WRITELN('insufficient memory') ;
      HALT                           ;
   END                               ;
   FOR index := 0 TO 8 DO NEW(uniform_data[index]) ;
   FILLCHAR(unif rm_data[8]^[715],108,0); { 714 ints, 108 bytes to null }
   percent d ne := 0;
   VAL(PARAMSTR(1),amplitude,temp_int);
   IF (temp int <> 0)
     OR (amplitude > 0)
       OR (amplitude < -99)
         OR (TRUNC(amplitude) <> amplitude)
           THEN
             BEGIN
               WRITE('negative integers between 0 and -99 are expected') ;
               HALT ;
             END ;
   amplitude   := 32767.0 * EXP((amplitude 20.0)*Ln(10.0)) ;
   frequency   := 2.0*PI*signal_frequency sampling_frequency ;

   FOR counter := 0 TO 131839 DO
   BEGIN
       IF (100*counter 131840) > percent_done THEN
       BEGIN
          WRITE(CHR(13),percent_done:2,'%');
          INC(percent_done);
       END;

       data := amplitude*SIN(frequency*(0.5+counter)) ;
       data := data+RANDOM 0.5                        ; { dither }
       index  :  counter SHR 14                       ;
       offset :  counter AND 16383                    ;
       uniform_data[index]^[offset] := ROUND(data)    ;
   END;

   ASSIGN (file out, paramstr(1)+'db in.int');
       REWRITE(file out);
           FOR index :  0 TO 7 DO
               BLOCKWRITE(file_out, uniform_data[index]^, 256);
               BLOCKWRITE(file_out, uniform_data[8]^, 12);
       CLOSE(file_out);

   RELEASE(heaporg);
   WRITE(CHR(7));
END.
```

NATSIGEN.PAS

```
PROGRAM uniformly_and_naturally_sampled_signal_generator;
{$N+,E+}
TYPE heaparray = ARRAY [0..49407] OF byte ;

CONST sampling_frequency = 352800   ; { 8x oversampling  }
      signal_frequency   = 1001     ; { Fc/Fv irrational }
      signal_amplitude   = 0.177828 ; { -15 dBFS (power) }
      no_of_bits         = 16       ; { variable limit ! }
      no_of_samples      = 16384    ; { 2x 48k,24bit bin }
      guard_slots        = 16       ; { used in nat only }

VAR   header_file, file_out      : file       ;
      uniform_data, natural_data : ^heaparray ;
      counter, no_of_slots       : longint    ;
      iterations, percent_done   : byte       ;
      amplitude, frequency, data : single     ;
      maxout, minout, position   : single     ;

BEGIN
   NEW(uniform_data);
   NEW(natural_data);
   percent_done := 0;

   ASSIGN (header_file, 'ch1-ch2.hed');
       RESET  (header_file);
           BLOCKREAD(header_file, uniform_data^,2);
           BLOCKREAD(header_file, natural_data^,2);
       CLOSE(header_file);

   no_of_slots := ROUND(EXP(no_of_bits*LN(2.0)))            ;
   amplitude   := signal_amplitude*(no_of_slots+1.0)/2.0    ;
   frequency   := 2.0*PI*signal_frequency/sampling_frequency ;

   FOR counter := 0 TO no_of_samples - 1 DO
   BEGIN
       IF (100*counter/no_of_samples)>percent_done THEN
       BEGIN
          INC(percent_done);
          WRITE(CHR(13),percent_done:2,'%');
       END;

       data := amplitude*SIN(frequency*(0.5+counter));

       uniform_data^[256+3*counter]  := HI(ROUND(data)); { calc. MS Byte }
       uniform_data^[256+3*counter+1]:= LO(ROUND(data)); { calc. LS Byte }
       uniform_data^[256+3*counter+2]:= 0              ; { insert 0 Byte }

       minout := -0.5                                  ; { variables for }
       maxout :=  0.5                                  ; { binary search }
       iterations := 0                                 ; { loop  counter }

       data := data + guard_slots*256;  { 8 bit dropped in NS; see below }

       WHILE iterations < no_of_bits DO                  { find int'sect }
       BEGIN
           INC(iterations);
           position := (minout + maxout)/2.0 ;
           data := 0.5*SIN(frequency*(0.5+position+counter))      ;
           IF data >= position THEN minout := position
                               ELSE maxout := position            ;
       END;

       data := 2.0*amplitude*position - guard_slots*256           ;

       natural_data^[256+3*counter]  := HI(ROUND(data)); { calc. MS Byte }
       natural_data^[256+3*counter+1]:= LO(ROUND(data)); { calc. LS Byte }
       natural_data^[256+3*counter+2]:= 0              ; { insert 0 Byte }
   END;

   ASSIGN (file_out, '16b8x1k.wav');
       REWRITE(file_out);
           BLOCKWRITE(file_out, uniform_data^, 386);
           BLOCKWRITE(file_out, natural_data^, 386);
       CLOSE(file_out);

   DISPOSE(uniform_data);
   DISPOSE(natural_data);
   WRITE(CHR(7));

END.
```

DITHER.PAS

```
PROGRAM dither_generator;
{$N+,B+,E+}
TYPE heaparray = ARRAY [0..49407] OF shortint ;

VAR   header_file, file_out : file               ;
      dither1, dither2      : ^heaparray         ;
      shiftreg              : longint            ;
      error, temp           : integer            ;
      counter               : word               ;
      tap1, tap2            : byte               ;
      pdf                   : ARRAY[0..8] OF real ;

BEGIN
   NEW(dither1)                          ;
   NEW(dither2)                          ;
   FILLCHAR(dither1^, 49408, 0)          ;
   FILLCHAR(dither2^, 49408, 0)          ;
   ASSIGN (header_file, 'ch1-ch2.hed')   ;
   RESET  (header_file)                  ;
      BLOCKREAD(header_file, dither1^,2) ;
      BLOCKREAD(header_file, dither2^,2) ;
   CLOSE(header_file)                    ;

   { 15 element shift register with feedback 15 XOR 14 }
   WRITELN('generating rectangular PDF dither')        ;
   shiftreg := 1 ;
   tap1     := 0 ;
   tap2     := 0 ;
   FOR counter := 0 TO 16383 DO
```

```
BEGIN
    IF (shiftreg AND 32768) <>0 THEN tap1:=1 ELSE tap1:=0 ;    { 2^15 }
    IF (shiftreg AND 16384) <>0 THEN tap2:=1 ELSE tap2:=0 ;    { 2^14 }
    shiftreg := shiftreg SHL 1                                 ;
    dither2^[257+3*counter]:=(tap1+tap2) MOD 2                 ;    { XOR! }
    shiftreg := shiftreg + dither2^[257+3*counter]             ;
END;

{ convertion to triangular PDF, HPF dither }
WRITELN('converting to triangular PDF HPF dither') ;
FOR counter := 16383 DOWNTO 1 DO
dither2^[257+3*counter] := dither2^[257+3*counter]
                         - dither2^[254+3*counter] +1 ; { nb offset }
{ 1st differencer }
WRITELN('differencing for 1st order loop')          ;
FOR counter := 16383 DOWNTO 1 DO
dither2^[257+3*counter] := dither2^[257+3*counter]
                         - dither2^[254+3*counter] ;
{ 2nd differencer }
WRITELN('differencing for 2nd order loop')             ;
FOR counter := 16383 DOWNTO 1 DO
dither2^[257+3*counter] := dither2^[257+3*counter]
                         - dither2^[254+3*counter] +4 ;
{ DSM }
WRITELN('bitstreaming to 3 values: 0,1,2') ;
error:=0                                   ;
counter:=257                               ;
REPEAT
    temp:=dither2^[counter]+error          ;
    error:=temp AND 3                      ;
    dither2^[counter]:=temp SHR 2          ;
    INC(counter,3)                         ;
UNTIL counter>=49408                       ;


{ PDF check one (on dither 2) }
WRITELN('calculating PDF of Dither II signal,') ;
WRITELN('   x    0     1     2')                ;
WRITE('Pr(x)')                                  ;
FOR counter := 0 to 2 DO pdf[counter] := 0.0    ;
FOR counter := 0 TO 16383 DO
BEGIN
IF (dither2^[257+3*counter] > 2)
   OR (dither2^[257+3*counter] < 0) THEN WRITE('out of range !') ;
pdf[dither2^[257+3*counter]]:=pdf[dither2^[257+3*counter]] +1    ;
END                                                              ;
FOR counter := 0 TO 2 DO WRITE(pdf[counter]/16384:6:3)           ;

{ 3rd differencer }
WRITELN,WRITELN('differencing for 3rd order loop') ;
FOR counter := 16383 DOWNTO 1 DO
dither1 [257+3*counter] := dither2^[257+3*counter]
                         - dither2^[254+3*counter] ;  { 1st loc'n = 0 }

{ 4th differencer and negation }
WRITELN('differencing for 4th order loop (with AP sign byte)') ;
FOR counter := 16383 DOWNTO 1 DO
BEGIN
dither1^[257+3*counter] :=  dither1^[254+3*counter]
                          - dither1^[257+3*counter]                ;
IF dither1^[257+3*counter] < 0 THEN dither1^[256+3*counter] := -1 ;
END                                                                ;

{ PDF check two (on dither 1) }
WRITELN('calculating PDF of Dither I signal,')                     ;
WRITELN('   x    -4    -3    -2    -1     0     1     2     3     4') ;
WRITE('Pr(x)')                                                     ;
FOR counter := 0 to 8 DO pdf[counter] := 0.0                       ;
FOR counter := 0 TO 16383 DO
BEGIN
IF (dither1^[257+3*counter] > 4)
   OR (dither1^[257+3*counter] < -4) THEN WRITE('out of range !')  ;
pdf[dither1^[257+3*counter]+4]:=pdf[dither1^[257+3*counter]+4] +1  ;
END                                                                ;
FOR counter := 0 TO 8 DO WRITE(pdf[counter]/16384:6:3)             ;

{ add two clock delay to dither II }
WRITELN;WRITELN('adding two clock cycle delay to Dither II signal.')  ;
FOR counter := 16383 DOWNTO 2 DO
                  dither2^[257+3*counter] := dither2^[251+3*counter] ;
dither2^[257] := 0 ;
dither2^[260] := 0 ;            { presume zero-pad before start point }

ASSIGN (file_out, 'dither.wav')        ;
REWRITE(file_out)                      ;
   BLOCKWRITE(file_out, dither1^, 386) ;
   BLOCKWRITE(file_out, dither2^, 386) ;
CLOSE(file_out)                        ;

RELEASE(heaporg) ;
WRITE(CHR(7))    ;

END.
```

A.1.2 : Floating point noise shapers

Two noise shapers are listed below; the first, 'NSX.PAS' uses an external ASCII file of coefficients to define the filter H(z) in a conventional multi-bit noise shaper. These are used to processes a '.wav' format input to produce a single channel '.nsx' format output (essentially the same format but with nulled LSBs in each data word). Compilation options are used to set the output wordlength. The second listing, 'DCNS4V4.PAS', is an example of a cascade structure noise shaper

using two quantisers. This assumes '.wav' format input and output, and uses separate data files for the signal and specially prepared dither (from DITHER.PAS).

NSX.PAS

```
PROGRAM for_16_to_8_bit_arbitrary_coefficient_noise_shaping;
{$N+}
TYPE heaparray = ARRAY [0..49407] OF byte ;

CONST no_of_samples = 16384 ;

VAR   io_file            : FILE                    ;
      coefficient_file   : TEXT                    ;
      uniform_data       : ^heaparray              ;
      data               : ARRAY [1..25] of REAL   ;
      coeff              : ARRAY [1..25] of REAL   ;
      output             : REAL                    ;
      counter1, counter2 : LONGINT                 ;
      percent_done       : BYTE                    ;

BEGIN
   IF PARAMCOUNT <> 1 THEN exit ;
   NEW(uniform_data)           ;

   ASSIGN (io_file, PARAMSTR(1)+'.wav')          ;
       RESET (io_file)                           ;
          BLOCKREAD(io_file, uniform_data^,386) ;
       CLOSE(io_file)                            ;

   ASSIGN (coefficient_file, 'nsfilter.asc')             ;
       RESET (coefficient_file)                          ;
           FOR counter1:= 1 TO 25 DO
           BEGIN
               READLN(coefficient_file, coeff[counter1]) ;
               data[counter1] := 0                       ;
           END                                           ;
       CLOSE(coefficient_file)                           ;

   counter1:= 256    ;
   percent_done := 0 ;

   WHILE counter1 <= 49405 DO
   BEGIN
       IF (100.0*(counter1-253)/(no_of_samples*3))>percent_done THEN
       BEGIN
          INC(percent_done)                  ;
          WRITE(CHR(13),percent_done:2,'%') ;
       END                                   ;

       output := 256 * uniform_data^[counter1]
                     + uniform_data^[counter1+1]
                     + 32768                                    ;
       FOR counter2 := 1 TO 25 DO
           output := output + coeff[counter2] * data[counter2] ;

       FOR counter2 := 25 DOWNTO 2 DO data[counter2] := data[counter2-1] ;
       data[1] := output-256*ROUND(output 256)                           ;
       uniform_data^[counter1]   := BYTE(ROUND(output/256)-128)          ;
       uniform_data^[counter1+1] := 0                                    ;
       INC(counter1,3)                                                   ;
   END                                                                   ;

   ASSIGN (io_file, PARAMSTR(1)+'.nsx')              ;
       REWRITE(io_file)                              ;
           BLOCKWRITE(io_file, uniform_data^, 386) ;
       CLOSE(io_file)                                ;
   DISPOSE(uniform_data)                             ;
   WRITE(CHR(7))                                     ;
END.
```

DCNS4V4.PAS'

```
PROGRAM fourth_order_cascaded_structure_16_to_8_bit_noise_shaper;
        { no_of_bits = 16! }
TYPE bytearray  = ARRAY [0..49407] OF byte              ;
     shortarray = ARRAY [0..49407] OF shortint          ;
CONST no_of_samples = 16384                             ;

VAR   file_in, file_out                  : file       ;
      data                               : ^bytearray  ;
      dither1, dither2                   : ^shortarray ;
      counter                            : longint     ;
      input1, input2, advanced_input2,
      feedback1, feedback2               : integer     ;
      output1,    output2,
      quantised1, quantised2,
      delayed1,   doubly_delayed1,
      hpf2,       doubly_hpf2            : shortint    ;
      sum1, sum2                         : word        ;
      error1, error2, delayed_error1,
      percent_done                       : byte        ;

BEGIN
   NEW(data)                ;
   NEW(dither1)             ;
   NEW(dither2)             ;
   FILLCHAR(data^,49408,0)  ;
   FILLCHAR(dither1^,49408,0) ;
   FILLCHAR(dither2^,49408,0) ;

   IF PARAMCOUNT <> 2 THEN exit ;

   ASSIGN (file_in, PARAMSTR(1)+'.wav') ;
   RESET (file_in)                      ;
      BLOCKREAD(file_in, data^,386)     ;
   CLOSE(file_in)                       ;
```

```
ASSIGN (file_in, PARAMSTR(2)+'.wav');
RESET (file_in)                      ;
   BLOCKREAD(file_in, dither1^,386) ;
   BLOCKREAD(file_in, dither2^,386) ;
CLOSE(file_in)                       ;

error1         := 0   ;
error2         := 0   ;
feedback1      := 0   ;
feedback2      := 0   ;
delayed1       := 0   ;
delayed_error1 := 0   ;
output1        := 0   ;
output2        := 0   ;
quantised1     := 0   ;
quantised2     := 0   ;
hpf2           := 0   ;
percent_done   := 0   ;
counter        := 256 ;

WHILE counter <= 49405 DO
BEGIN
    IF (100*(counter-253)/(no_of_samples*3))>percent_done THEN
    BEGIN
       INC(percent_done)              ;
       WRITE(CHR(13),percent_done:2,'%') ;
    END                               ;

    doubly_delayed1  := delayed1        ;  { clock output 1 2nd delay }
    delayed1         := output1         ;  { clock output 1 1st delay }
    output1          := quantised1      ;  { clock quantiser 1 output }
    advanced_input2  := delayed_error1  ;  { clock input2 into ons2.. }
    delayed_error1   := error1          ;  { clock error1 out of ons1 }

                                           { begin first noise shaper }
    input1 := 256 * data^[counter]
              +    data^[counter+1]
              + dither1^[counter+1]
              + 32768                  ;      { word offset : 16 bit }
    sum1           := input1 + feedback1    ;
    quantised1     := sum1 DIV 256 - 128    ;      { word offset removed. }
    feedback1      := - error1              ; { temporary use of variable }
    error1         := sum1 MOD 256          ;
    feedback1      := feedback1 + 2*error1  ;
                                                        { start hpfs }
    doubly_hpf2 := - hpf2                   ; { temporary use of variable }
    hpf2        := - output2                ; { temporary use of variable }
    output2     := quantised2               ;  { clock quantiser 2 output }

                                           { begin second noise shaper }
    input2      := advanced_input2
                 + dither2^[counter+1]   ;
    sum2        := input2 + feedback2    ;
    quantised2  := sum2 DIV 256          ;
    feedback2   := - error2              ; { temporary use of variable }
    error2      := sum2 MOD 256          ;
    feedback2   := feedback2 + 2*error2  ;
                                                      { finish hpfs }
    hpf2 := hpf2 + output2               ;
    doubly_hpf2 := doubly_hpf2 + hpf2    ;

    data^[counter] := doubly_hpf2 + doubly_delayed1 ; { output answer }
    INC(counter)        ;
    data^[counter] := 0 ;                          { insert zero byte }
    INC(counter,2)      ;
    END                 ;

ASSIGN (file_out, PARAMSTR(1)+'.cs4') ;
REWRITE(file_out)                     ;
    BLOCKWRITE(file_out, data^, 386)  ;
CLOSE(file_out)                       ;

RELEASE(heaporg) ;
WRITE(CHR(7))    ;

END.
```

A.1.3 : Feedforward noise shaper

One example of a feedforward noise shaper is presented below, 'QENS.FOR'. This operates by quantising the difference between input and output of a noise shaper, using this as an estimate of the shaped error that has been added, and where large enough to be spread more evenly a second filter is used to feedforward a re-shaped error which reduces the overall noise power gain.

QENS.FOR

```
      PROGRAM FLOATING_POINT_QENS
      EXTERNAL RAN2
C
      IMPLICIT NONE
      INTRINSIC NINT, AINT
C
C        variable explanation
C        --------------------
C     fb : feedback data values
C     ff : feedforward data values
C     kf : feedforward gain
C     kr : recursive gain
C     H1 : feedforward coefficients
C     H2 : feedback coefficients
C     do : delayed output values
C     preset(1) allows FF filtering if set to 1
```

```
C      preset(2) adds 1 bit rectangular pdf dither if set to 1
C
       INTEGER*4 INPUT, OUTPUT, I, SEED, DO
       REAL FB, FF, H1, H2, S
       DOUBLE PRECISION RAN2, DITHER, PRESET
       DIMENSION FB(20), FF(20), H1(20), H2(20), DO(2), PRESET(2)
       DO 1 I=1,20
          FB(I)=0.0
          FF(I)=0.0
          H1(I)=0.0
          H2(I)=0.0
1      CONTINUE
       DO(1)=0
       DO(2)=0
       SEED=-6              ! initialise random No. generator
       DITHER=RAN2(SEED)    ! don't update the returned value
       INPUT=0
C start debug
       OPEN(UNIT=15,FILE='qenspar',STATUS='UNKNOWN',ERR=6)
       READ(15,*) PRESET(1),PRESET(2)
       CLOSE(UNIT=15)
C 8x16b8 (11th order) : this has a gain of 7.77
C      H2(0)=-1.0
       H2(1)=2.15620339
       H2(2)=-0.541145028
       H2(3)=-1.13821398
       H2(4)=0.252112901
       H2(5)=-0.166321250
       H2(6)=0.480820741
       H2(7)=0.167501873
       H2(8)=0.110407013
       H2(9)=-0.356660342
       H2(10)=-0.130069931
       H2(11)=0.161053596
C -13 from initnsopt (11th order) : this has a gain of 15.93
C      H1(-1)=-1            ! Forward time coeffs must be ints and must
C      H1(0)=3              ! be added after the noise o/p is measured.
       H1(1)=-2.26244529
       H1(2)=-0.517323895
       H1(3)=0.326091441
       H1(4)=0.536745695
       H1(5)=0.185858203
       H1(6)=0.0519153763
       H1(7)=-0.251861825
       H1(8)=-0.163171083
       H1(9)=-0.0497250203
       H1(10)=0.148383235
C stop debug
2      CONTINUE                                  ! start of loop
       READ(5,*,END=6) INPUT
       S=INPUT
       DO 3 I=1,20
          S=S+H2(I)*FB(I)-H1(I)*FF(I)            ! both filters applied here
3      CONTINUE
       OUTPUT=256*NINT(S/256)                    ! NINT rounds & o ps INTEGER
       DO 4 I=20,2,-1
          FB(I)=FB(I-1)
          FF(I)=FF(I-1)
4      CONTINUE
       DITHER=RAN2(SEED)-0.5                     ! -0.5 < DITHER < +0.5
       FB(1)=S-OUTPUT+DITHER*PRESET(2)
       FF(1)=OUTPUT-INPUT                   ! I/O difference (in i p LSBs)
       FF(1)=FF(1)/256.0                    ! I/O difference (in o p LSBs)
       FF(1)=NINT(FF(1)/3)                  ! round MOD 3 (in o p LSBs)
       FF(1)=256*FF(1)                      ! quantised I O error (i p LSB)
       FF(1)=FF(1)*PRESET(1)                   ! optionally suppress
       DO(2)=DO(1)+1*NINT(FF(1))               ! H1(-1) hardwired
       DO(1)=OUTPUT-3*NINT(FF(1))              ! H1(0) hardwired
       WRITE(6,'(1E20.10)') DO(2)/32767.0      ! exp, scale to +/- 0.5
       GOTO 2                                  ! input from sine.f was
6      CONTINUE                                ! ranged +/- 32767
       STOP

       END
```

A.1.3 : Open Loop Pre-compensators

Two examples of open loop pre-compensation routines are listed below, 'WAPWM.PAS' and 'LANR.PAS'. Both operate by approximating natural sampling, the first using a weighted average of adjacent uniformly sampled data (linear interpolation as used by MEL91), the second using a third order Lagrangian polynomial approximation of the signal with one Newton-Raphson iteration to find the natural sampling instant. Both can handle up to 24 bit resolution signals and use '.wav' format input and output.

WAPWM.PAS

```
PROGRAM  to_find_weighted_averaged_values_from_uniformly_sampled_ones  ;

{ revise compiler directives for debugging }
{$A+,B-,D-,E-,F-,I+,L-,N+,O-,R-,S-,V+}
{$M 16384,0,655360}

   TYPE heaparray = ARRAY [0..8191] OF real                    ;
        ap_format = ARRAY [0..49407] OF byte                   ;

   VAR   file_in, file_out            : file                   ;
         counter, temp_long           : longint                ;
         index, offset, percent_done  : word                   ;
         range_in, range_out,
```

```
      offset_in, offset_out,
      guard_band,
      x0, x1, x                    : real                        ;
      data                         : ARRAY[0..1] OF ^heaparray ;
      io, bin_nat_data             : ^sp_format                  ;

BEGIN
   IF PARAMCOUNT <> 1 THEN
   BEGIN
       WRITELN('input filename expected (.wav assumed)')          ;
       HALT                                                       ;
   END                                                            ;

   IF MEMAVAIL < 212128 THEN
   BEGIN
       WRITELN ('insufficient memory') ;
       HALT                            ;
   END                                 ;

   NEW(data[0])                              ;
   NEW(data[1])                              ;
   NEW(io)                                   ;
   NEW(bin_nat_data)                         ;

   ASSIGN (file_in, PARAMSTR(1)+'.wav')      ;
   RESET (file_in)                           ;
   BLOCKREAD(file_in, io^, 386)              ;
   BLOCKREAD(file_in, bin_nat_data^, 386)    ;
   CLOSE(file_in)                            ;

   FOR counter := 0 TO 16383 DO
   BEGIN
       index := counter SHR 13 ;
       offset := counter AND 8191 ;
{ ** EITHER ** , for 24 bit data : }
       data[index]^[offset] :=  256.0*SHORTINT(io^[256+3*counter])
                              +                io^[257+3*counter]
                              +                io^[258+3*counter] /256.0 ;
{ **** OR **** , for 16 bit data :
      data[index]^[offset] :=  256.0*SHORTINT(io^[257+3*counter])
                              +                io^[258+3*counter] ; }
{ debug : WRITELN(data[index]^[offset]:5:3); }
   END;

   RandSeed       := 1       ;
   percent_done   := 0       ;
   range_in       := 65536.0 ;
   range_out      := 73728.0 ;                  { 88% modulation depth }
   offset_in      := 32768.0 ;
   offset_out     := 36864.0 ;       { mid range, add guard band later }
   guard_band     := 4096.0  ;       { 16 cycles after dropping 8 bits }

{ setup initial input }
   x0 := (data[0]^[0]+offset_in)/range_in ;            { normalised }

   FOR counter := 0 TO 16383 DO
   BEGIN
       IF (100*(counter)/(16383)) > percent_done THEN
       BEGIN
           INC(percent_done)                 ;
           WRITE(CHR(13),percent_done:2,'%') ;
       END                                   ;

       index := counter SHR 13 ;
       offset := counter AND 8191 ;

     { read in a new x(t) }
       x1 := (data[index]^[offset]+offset_in)/range_in ;

     { evaluate weighted average value }
       x    := x0 (1-(x1-x0))  ;

     { rescale the output }
       x := (x * range_out)-offset_out  ;

       IF (x < guard_band-offset_out ) THEN
       BEGIN
           WRITELN('underflow!..inadequate guard band, aborting') ;
           HALT;
       END;

       x := x-guard_band                       ;
     { x := x*256                              ;     16 bits -> 24 bits }
       temp_long := ROUND(x+RANDOM-0.5) ;          { dither & round off }
       io^[258+3*counter] := 0      { temp_long AND 255 ; remove brackets
       temp_long := temp_long SHR 8 }                    ; { for 24 bits }
       io^[257+3*counter] := temp_long AND 255 ;
       temp_long := temp_long SHR 8 ;
       io^[256+3*counter] := temp_long AND 255 ;

     { clock the data }
       x0 := x1 ;

   END ;

   ASSIGN (file_out, PARAMSTR(1)+'$.wav')    ;
   REWRITE(file_out)                         ;
   BLOCKWRITE(file_out, io^, 386)            ;
   BLOCKWRITE(file_out, bin_nat_data^, 386) ;
   CLOSE(file_out)                           ;

   RELEASE(heaporg) ;
   WRITE(CHR(7))    ;

END.
```

LANR.PAS

```
PROGRAM  to_find_naturally_sampled_values_from_uniformly_sampled_ones  ;
     {  using third order Lagrange ploynomial signal approximation &  }
     {  one Newton Raphson iteration from a weighted average guess .  }

{revise compiler directives for debugging!}
{$A+,B-,D-,E-,F-,I+,L-,N+,O-,R-,S-,V+}
{$M 16384,0,655360}

   TYPE heaparray = ARRAY [0..8191] OF real                    ;
        ap_format = ARRAY [0..49407] OF byte                   ;

   VAR   file_in, file_out           : file                    ;
         counter, temp_long          : longint                 ;
         index, offset, percent_done : word                    ;
         range_in, range_out,
         offset_in, offset_out,
         A, B, x, P4, dP4, guard_band : real                   ;
         s,y                         : ARRAY[1..4] OF real     ;
         data                        : ARRAY[0..1] OF ^heaparray ;
         io, bin_nat_data            : ^ap_format              ;

   BEGIN
      IF PARAMCOUNT <> 1 THEN
      BEGIN
          WRITELN('input filename expected (.wav assumed)')          ;
          HALT                                                       ;
      END                                                            ;

      IF MEMAVAIL < 197320 THEN
      BEGIN
          WRITELN ('insufficient memory') ;
          HALT                            ;
      END                                 ;

      NEW(data[0])                        ;
      NEW(data[1])                        ;
      NEW(io)                             ;
      NEW(bin_nat_data)                   ;

      ASSIGN (file_in, PARAMSTR(1)+'.wav')  ;
      RESET (file_in)                       ;
      BLOCKREAD(file_in, io^, 386)          ;
      BLOCKREAD(file_in, bin_nat_data^, 386) ;
      CLOSE(file_in)                        ;

      FOR counter := 0 TO 16383 DO
      BEGIN
          index := counter SHR 13 ;
          offset := counter AND 8191 ;
    { ** EITHER ** , for 24 bit data : }
          data[index]^[offset] :=  256.0*SHORTINT(io^[256+3*counter])
                                 +                io^[257+3*counter]
                                 +                io^[258+3*counter]  256.0 ;
    { **** OR **** , for 16 bit data :
          data[index]^[offset] :=  256.0*SHORTINT(io^[257+3*counter])
                                 +                io^[258+3*counter] ; }
    { debug : WRITELN(data[index]^[offset]:5:3); }
      END;

      RandSeed       := 1       ;
      percent_done   := 0       ;
      range_in       := 65536.0 ;
      range_out      := 73728.0 ;                  { 88% modulation depth }
      offset_in      := 32768.0 ;
      offset_out     := 36864.0 ;       { mid range, add guard band later }
      guard_band     := 4096.0  ;       { 16 cycles after dropping 8 bits }

    { setup first data set }
      s[1] := (data[0]^[0]+offset_in)/range_in ;            { normalised }
      s[2] := s[1]       ;
      s[3] := s[1]       ;    { use first point duplicates to start with }
      y[1] := s[1] + 1.0 ;
      y[2] := s[2]       ;
      y[3] := s[3] - 1.0 ;

      FOR counter := 0 TO 16383 DO
      BEGIN
          IF (100*(counter)/(16383)) > percent_done THEN
          BEGIN
              INC(percent_done)                 ;
              WRITE(CHR(13),percent_done:2,'%') ;
          END                                   ;

          index := counter SHR 13 ;
          offset := counter AND 8191 ;

        { read in a new value }
          s[4] := (data[index]^[offset]+offset_in)/range_in ;
          y[4] := s[4] - 2.0 ;

        { evaluate weighted average first guess }
          A    := 1 - s[3] ;
          B    := s[2]     ;
          x    := B/(A+B)  ;

        { evaluate P4, the Lagrange cubic polynomial }
          P4   := ( y[4] - 3*y[3] + 3*y[2] - y[1] ) * x ;     { cubic term }
          P4   := ( P4 + 3*y[3] - 6*y[2] + 3*y[1] ) * x ;     { quadratic }
          P4   := ( P4 - y[4] + 6*y[3] - 3*y[2] - 2*y[1] ) * x ;  { linear }
          P4   := ( P4 +6*y[2] )/6 ;                           { constant }

        { evaluate dP4, the polynomial derivative }
          dP4  := ( 3*y[4] - 9*y[3] + 9*y[2] - 3*y[1] ) * x ;  { quadratic }
          dP4  := ( dP4 + 6*y[3] - 12*y[2] + 6*y[1] ) * x ;       { linear }
          dP4  := ( dP4 - y[4] + 6*y[3] - 3*y[2] - 2*y[1] )/6 ; { constant }

          x := x - P4/dP4 ;                  { one Newton Raphson iteration }

          x := (x * range_out)-offset_out  ;          { rescale the output }
          IF (x < guard_band-offset_out ) THEN
          BEGIN
```

```
        WRITELN('underflow!..inadequate guard band, aborting') ;
        HALT;
      END;
      x := x-guard_band                          ;
    { x := x*256                                 ;     16 bits -> 24 bits }
      temp_long := ROUND(x+RANDOM-0.5) ;          ( dither & round off )
      io^[258+3*counter] := 0      { temp_long AND 255 ; remove brackets
      temp_long := temp_long SHR 8 }                   ; { for 24 bits }
      io^[257+3*counter] := temp_long AND 255 ;
      temp_long := temp_long SHR 8 ;
      io^[256+3*counter] := temp_long AND 255 ;

      FOR index := 2 TO 4 DO
      BEGIN
          s[index-1] := s[index]     ;
          y[index-1] := y[index]+1.0 ;
      END ;
    END ;

    ASSIGN (file_out, PARAMSTR(1)+'$.wav')    ;
    REWRITE(file_out)                         ;
    BLOCKWRITE(file_out, io^, 386)            ;
    BLOCKWRITE(file_out, bin_nat_data^, 386) ;
    CLOSE(file_out)                           ;

    RELEASE(heaporg) ;
    WRITE(CHR(7))    ;

END.
```

A.1.3 : Closed Loop Simulators

Two examples of closed loop systems to linearise PWM are presented below. The first, 'SIPWM.PAS' uses a $\Sigma\Delta$ structure (output signal feedback with IIR G{z} in the signal path), but replaces the quantiser with a pulse width modulator. 'NSPWM.PAS' is similar but uses a a noise shaper structure with a decimated error feedback signal. Both are unstable for all but very high oversampling ratios, hence ramp, exponential and small signal startup is included to assess the onset of instability, along with limiters and overload detection to prevent processing of meaningless data.

SIPWM.PAS

```
PROGRAM to_simulate_a_pwm_instead_of_a_quantiser_in_a_sigma_delta  ;

{$A+,B-,D+,E-,F-,I+,L+,N+,O-,R+,S+,V+}
{$M 16384,0,655360}

{ beware! sigma delta modulators suffer from overflow with b < 4 }

TYPE heaparray = ARRAY [0..16383] OF integer                      ;

VAR   file_in, file_out              : FILE                       ;
      counter                        : LONGINT                    ;
      input, first_val, int_out, bits : INTEGER                   ;
      index, offset, i               : WORD                       ;
      percent_done                   : BYTE                       ;
      dither, range, maxval          : EXTENDED                   ;
      sum, output,
      forward_coeff, backward_coeff  : ARRAY [0..4] OF EXTENDED   ;
      data                           : ARRAY [0..8] OF ^heaparray ;

BEGIN
   IF PARAMCOUNT <> 3 THEN
   BEGIN
       WRITELN('usage: filename (?in.int), dither (y/n), No. of bits')  ;
       HALT                                                             ;
   END                                                                  ;

   IF MEMAVAIL < 294912 THEN
   BEGIN
       WRITELN ('insufficient memory') ;
       HALT                            ;
   END                                 ;

   VAL(PARAMSTR(3),bits,int_out)              ;
   IF (int_out <> 0) THEN
   BEGIN
      WRITELN('error in parameter 3')         ;
      HALT                                    ;
   END                                        ;
   range := 65536.                            ;
   FOR i := 1 TO bits DO range := range / 2. ;
   bits := 16 - bits                          ;
   FOR index := 0 TO 8 DO NEW(data[index]) ;
   ASSIGN (file_in, PARAMSTR(1)+'in.int')  ;
   RESET (file_in)                         ;
   FOR index := 0 TO 7 DO BLOCKREAD(file_in, data[index]^,256) ;
   BLOCKREAD(file_in, data[8]^,12)         ;
   CLOSE(file_in)                          ;
   FOR i := 1 TO 4 DO
   BEGIN
       sum[i]    := 0.0 ;
       output[i] := 0.0 ;
   END                  ;

   forward_coeff[0] :=  1.0 { ?, unused ! } ;
   forward_coeff[1] :=  1.0     ;
   forward_coeff[2] := -2.4541  ;
   forward_coeff[3] :=  2.042046 ;
   forward_coeff[4] := -0.565514 ;
```

```
        backward_coeff[0] :=  1.0 ( ?, unused ! ) ;
        backward_coeff[1] := -4.0 ; { -3.9914352 ;}
        backward_coeff[2] :=  6.0 ; {  5.9758155 ;}
        backward_coeff[3] := -4.0 ; { -3.9773245 ;}
        backward_coeff[4] :=  1.0 ; {  0.9929441 ;}

        RandSeed     := 1           ;
        first_val    := data[0]^[0] ;
        maxval       := 0.0         ;
        dither       := 0.0         ;
        percent_done := 0           ;

{ Percent done = 0 is used to indicate that the modulator is still in a
  transient state. When the input returns to a value near the start point
  (a periodic signal is assumed) and at least 5000 samples have been
  processed (to allow for impulse responses to decay), the modulator is
  re-started, percent done is moved off zero and the output is connected. }

        FOR counter := 0 TO 131839 DO
        BEGIN
            index  := counter SHR 14       ;                  {   array    }
            offset := counter AND 16383    ;                  { references }

            input  := data[index]^[offset] ;                  { new input  }

            IF (percent_done=0)
              AND (counter > 5000)
                AND (input=first_val)
                  THEN                                        {  smart   }
                    BEGIN                                     { restart. }
                       counter := 0      ;
                       index   := 0      ;
                       offset  := 0      ;
                       INC(percent_done) ;
                    END                  ;

            IF (100*(counter/131840) > percent_done)
              AND (percent_done <> 0)                           { progress  }
                THEN                                            { indicator }
                  BEGIN
                    INC(percent_done)                  ;
                    WRITE(CHR(13),percent_done:2,'%') ;
                  END                                 ;

            output[0] := 0.0                                  ;
            FOR i := 1 TO 4 DO
            BEGIN                                               { recursive  }
                output[0] := output[0]                          {    LPF     }
                           + forward_coeff[i] * sum[i]          {  separate  }
                           - backward_coeff[i] * output[i]    ; { multiplies }
            END                                               ; { -ve back ? }

            IF (PARAMSTR(2) = 'y') OR (PARAMSTR(2) = 'Y') THEN
               dither := RANDOM - 0.5                         ; { new dither }

            IF (ABS(output[0]+dither) > maxval) THEN
            BEGIN
               maxval := ABS(output[0]+dither)                ; {  overload  }
               IF (ROUND(maxval/range) SHL bits > 32767) THEN   {   check.   }
               BEGIN
                  WRITE('the output''s overflowed')           ;
                  HALT                                        ;
               END                                            ;
            END                                               ;

            int_out := ROUND((output[0]+dither)/range) SHL bits;{ quantiser. }
            sum[0]  := input - int_out                        ; { next 'sum' }
            FOR i := 4 DOWNTO 1 DO                              {            }
            BEGIN                                               {            }
                output[i] := output[i-1]                      ; { clocking.. }
                   sum[i] :=    sum[i-1]                      ; {            }
            END                                               ; {            }

            IF (percent_done <> 0) THEN                         { ramp start }
{            BEGIN
              IF (counter < 1000) THEN
                 data[index]^[offset] := ROUND(int_out * counter/1000.0)
              ELSE) data[index]^[offset] := int_out ;           { overwrite! }
{            END ;}
        END ;

        ASSIGN (file_out, PARAMSTR(1)+'sj.int') ;
        REWRITE(file_out)                       ;
        FOR index := 0 TO 7 DO BLOCKWRITE(file_out, data[index]^, 256) ;
        BLOCKWRITE(file_out, data[8], 12)       ;
        CLOSE(file_out)                         ;

        RELEASE(heaporg)                                ;
        WRITE(CHR(13),CHR(10),'maxval =',maxval:8:1,CHR(7)) ;
    END.
```

NSPWM.PAS

```
PROGRAM for_noise_shaped_pwm_using_decimated_output_as_the_feedback ; { $N+ }
TYPE heaparray = ARRAY [0..49407] OF BYTE     ;
     realarray = ARRAY [1..18] OF REAL        ;

{ global variables }
VAR  counter      : WORD                      ;
     input        : INTEGER                   ;
     sum, error   : REAL                      ;
     output       : SHORTINT                  ;
     IIR_data     : ARRAY[1..7,1..3] OF REAL  ;
     Hns_data     : ARRAY[1..30] OF REAL      ;
     G_data       : ARRAY[1..2] OF REAL       ;
     F_data       : ARRAY[1..2] OF REAL       ;
     data         : ^heaparray                ;
     internal     : ^realarray                ;
     Hns          : ARRAY[1..30] OF REAL      ;
     L            : ARRAY[0..35] OF REAL      ;
     K,B,C,D      : ARRAY[1..6]  OF REAL      ;
     percent_done : BYTE                      ;
```

```
{ procedures }

PROCEDURE GET_DATA                          ;
VAR io_file : FILE                          ;
BEGIN
    ASSIGN (io_file, PARAMSTR(1)+'.wav')    ;
        RESET(io_file)                      ;
        NEW(data)                           ;
        BLOCKREAD(io_file, data^,386)       ;
    CLOSE(io_file)                          ;
END                                         ;

PROCEDURE SAVE_DATA                         ;
VAR io_file : FILE                          ;
BEGIN
    ASSIGN (io_file, PARAMSTR(1)+'.nsp')    ;
        REWRITE(io_file)                    ;
        BLOCKWRITE(io_file, data^, 386)     ;
        DISPOSE(data)                       ;
    CLOSE(io_file)                          ;
    WRITE(CHR(7))                           ;
END                                         ;

PROCEDURE GET FILTERS                                                   ;
VAR coefficient file : TEXT                                             ;
    counter : WORD                                                      ;
BEGIN
    ASSIGN (coefficient_file, 'Hns.asc')                                ;
        RESET (coefficient_file)                                        ;
        FOR counter := 1 TO 30 DO READLN(coefficient_file, Hns[counter]) ;
    CLOSE(coefficient_file)                                             ;
    ASSIGN (coefficient_file, 'f1.asc')                                 ;
        RESET (coefficient_file)                                        ;
        FOR counter:= 0 TO 31 DO READLN(coefficient_file,L[counter])    ;
    CLOSE(coefficient_file)                                             ;
    ASSIGN (coefficient_file, 'f2.asc')                                 ;
        RESET (coefficient_file)                                        ;
        FOR counter:= 1 TO 6 DO
        BEGIN
            READLN(coefficient_file, K[counter])                        ;
            READLN(coefficient_file, B[counter])                        ;
            READLN(coefficient_file, C[counter])                        ;
            READLN(coefficient_file, D[counter])                        ;
        END                                                             ;
    CLOSE(coefficient_file)                                             ;
END                                                                     ;

PROCEDURE CLEAR_FILTER_MEMORIES                         ;
VAR counter : WORD                                      ;
BEGIN
    G_data[1] := 0.0                                    ;
    G_data[2] := 0.0                                    ;
    F_data[1] := 0.0                                    ;
    F data[2] := 0.0                                    ;
    IIR_data[7,2] := 0.0                                ;
    IIR_data[7,3] := 0.0                                ;
    FOR counter := 1 TO 30 DO Hns_data[counter] := 0.0  ;
END                                                     ;

PROCEDURE INITIALISE_IIR                                        ;
VAR init_file : TEXT                                            ;
BEGIN
    ASSIGN (init_file, '16b8x1k.ini')                           ;
    RESET (init_file)                                           ;
    FOR counter := 1 TO 7 DO READLN(init_file,IIR_data[counter,1]) ;
    FOR counter := 1 TO 6 DO READLN(init_file,IIR_data[counter,2]) ;
    FOR counter := 1 TO 6 DO READLN(init_file,IIR_data[counter,3]) ;
    CLOSE(init_file)                                            ;
END                                                             ;

PROCEDURE CHECK_PROGRESS (16384 samples)  ;
BEGIN
    IF (100.0*(counter-253)/(16384*3))>percent_done THEN
    BEGIN
        INC(percent_done)                 ;
        WRITE(CHR(13),percent_done:2,'%') ;
    END                                   ;
END                                       ;

{ functions }

FUNCTION F1(data:SHORTINT):pointer          ;
{ decimates by 16, assuming a run length coded, TEPWM input; it refers to  }
{ a table of convolution answers stored in the (global) array L, of reals, }
{ & assumes there's a 32 cycle guardband, 50% high, 50% low, at the start. }
VAR counter, temp_word : WORD               ;
BEGIN
    internal^[1] := L[16]                   ;                   { 16 overlap }
    counter := 1                            ;
    temp_word := data+144                   ;
    WHILE temp_word >= 31 DO
    BEGIN
        INC(counter)                        ;
        internal^[counter]:= L[31]          ;                   { 31 overlap }
        DEC(temp_word,16)                   ;
    END                                     ;
    INC(counter)                            ;
    internal^[counter] := L[temp_word]      ;       { 30 >= overlap >= 15 }
    DEC(temp_word,16)                       ;
    IF temp_word > 0 THEN
    BEGIN
        INC(counter)                        ;
        internal^[counter] := L[temp_word]  ;
    END                                     ;
    WHILE (counter) < 18 DO
    BEGIN
        INC(counter)                        ;
        internal^[counter]:= L[0]           ;                   { 0 overlap }
    END                                     ;
    F1 := internal                          ;
END                                         ;
```

```
FUNCTION F2(dummy:pointer) : REAL                    ;
{ decimates by 18, using a 12th order IIR configured as a series of biquad }
{ sections; it requires coefficients from the (global) arrays K,B,C,D, and }
{ interim answers are stored in the calling functions array 'IIR_data'.    }
{ nb: for effective operation this filter should have a group delay in the }
{ passband of 35 input cycles, so that having added the single delay from  }
{ the preceeding filter (18 @ 18x) the overall delay is two output samples }
{ which can be accomodated for by an optimised noise shaing filter, Hns.   }
VAR counter, n : BYTE                                ;
BEGIN
    FOR counter := 1 TO 18 DO                      { use all of the data.. }
    BEGIN
        IIR_data[1,1] := internal^[counter]          ;
        FOR n := 1 TO 6 DO                         { in all of the biquads }
        BEGIN
            IIR_data[n,1]    := K[n]* IIR_data[n,1]        { recursive part }
                              - C[n]* IIR_data[n,2]
                               - D[n]* IIR_data[n,3] ;
            IIR_data[n+1,1] := IIR_data[n,1]          { non-recursive part }
                             + B[n]* IIR_data[n,2]
                              + IIR_data[n,3]        ;
            IIR_data[n,3] := IIR_data[n,2]           ;             { shift }
            IIR_data[n,2] := IIR_data[n,1]           ;             { shift }
        END                                          ;
    END                                              ;          {TEST!}
{    F2 := 8.106212079E+4*(IIR_data[7,1] - 3.26660974E-3) ;  output weight }
END                                                  ;

FUNCTION F(data:SHORTINT) : REAL ;
{ decimates by 288 times in two filters : F1 (FIR lookup) & F2 (6 biquads) }
BEGIN
    internal := F1(data)     ; { internal's a dummy variable; it points at }
           F := F2(internal) ; { array L of FIR lookup-table convolutions. }

{ TEST! }  F := F_data[2];          { F bypass }
   F_data[2] := F_data[1];
   F_data[1] := 256.0 * data ;
END                          ;

FUNCTION G(data:REAL):REAL ;
{ to apply filter 'G' to global 'sum'... here G's all pass, 2 cycle delay. }
BEGIN
    G := G_data[2]         ;
    G_data[2] := G_data[1] ;
    G_data[1] := data      ;
END                        ;

FUNCTION H(error:REAL):REAL                                              ;
{ operates filter 'H' on error & Hns_data with global coefficients, 'Hns'; }
{ error, the current error, must be prepared before entry, Hns_data[1..30] }
{ are the past errors, & the filter output is returned in H. }
VAR counter        : WORD                                                ;
    filtered_error : REAL                                                ;
BEGIN
    FOR counter := 30 DOWNTO 2 DO Hns_data[counter] := Hns_data[counter-1]  ;
    Hns_data[1] := error    { clock errors by one, include new error }     ;
    filtered_error := 0.0       { prepare to accumulate output }           ;
    FOR counter := 1 TO 30 DO            { & convolve }
    BEGIN
        filtered_error := filtered_error + Hns[counter] * Hns_data[counter] ;
    END                                                                  ;
    H := filtered_error                                                  ;
END                                                                      ;

FUNCTION QPWM(input:REAL) : SHORTINT   ;
BEGIN
    IF ABS(input) >= 32640 THEN
    BEGIN
        WRITE(CHR(13),CHR(27),'[5;1;31mLIMITING !',CHR(27),'[0m');
        IF input > 0 THEN QPWM :=  127
                     ELSE QPWM := -127 ;
    END
    ELSE QPWM := ROUND(input/256.0)    ;
END                                    ;


{ main program }

BEGIN
    IF PARAMCOUNT <> 1 THEN exit ;
    GET_FILTERS                  ;
    GET_DATA                     ;
    CLEAR_FILTER_MEMORIES        ;
    INITIALISE_IIR               ;

    NEW(internal)     ;
    error := 0.0      ;
    counter := 256    ;
    percent_done := 0 ;

    WHILE counter <= 49405 DO
    BEGIN                                    { - 2^15 < input < + (2^15 -1) }
        input := 256 * data^[counter] + data^[counter+1]    ;
        sum := input + H(error)                             ;
        output := QPWM(sum)                                 ;
        error := G(sum) - F(output)                         ;
        data^[counter] := output                            ;
        data^[counter+1] := 0                               ;
        CHECK_PROGRESS                                      ;
        INC(counter,3)                                      ;
    END                                                     ;
    SAVE_DATA                                               ;
END.
```

Appendix 2 : Analysis & display software.

A.2.1 : FIR Decimator : 'DECIMATE.600'

Shown below is a example of the fast decimation software, studied in chapter three, which prepares TEPWM bitstream data for spectral analysis over the audio band. In this example, sample rate increase introduced by the PWM is removed leaving a 24 bit integer result in a '.wav' format file for FFT analysis in another module. An external coefficient set is required for the last (and largest) filter but the other filters are compiled-in. Appropriate header files for the '.wav' files are also required in the current directory.

DECIMATE.600

```
PROGRAM to_decimate_a_run_length_code_file_by_288x_and_output_an_AP_wav_file;

        { compile to use 80287 coprocessor without range checking }
        {$A+,B-,D-,E-,F-,I+,L-,N+,O-,R-,S-,V-}  {$M 16384,0,655360}

{   The input is assumed to be rlc (highs only), 8 bit quantised, expressed
    in 2's complement, and is offset by 16 for a 288 unit/frame, (where one
    frame is the period of the oversampled signal ie: one input datum). 162
    spare samples should be provided out of filter 4 to accomodate convolu-
    tion end effects in filter 5 (163 taps, and decimating by 2). For this,
    filter 4 requires 350 spare samples, (162 * 2, for filter 5's ends, +27
    for the filter 4's end effects,-1 since it's decimated by 2). Similarly
    filter 3 will have to be given 714 spare samples (350*2+15-1), filter 2
    will need 1434 spare (714*2+7-1), so filter 1 must output an extra 1434
    modulated, combed samples, requiring 25830 'samples' (1434*18+35-17) or
    at least 90 modulated samples extra. Since this is read from file space
    must be provided to the nearest block so 128 extra samples will have to
    accomodated.  18KB is provided, for indexing simplicity, but only 16474
    input points are modulated, (and are zero padded if unused) .

    filter 1: A second order comb, 35 coefficients long, with 18 coincident
    pole pairs, equispaced in frequency from 0 to Fs.The coefficients are a
    convolution of two rectangles of unit height, 18 samples long, which is
    calculated by the taking an arithmetic series (with unit increment from
    1 to the position value, max. 18) and symmetric about the 18th (centre)
    position. ie:1,3,6,10...153,171,153...10,6,3,1. To speed up convolution
    the accumulation for each case of the 1 - 35 sample overlap between the
    filter and the input is calculated beforehand and referenced by array.

    filter 2: A 7 tap, half-band LPFIR, with integer coefficients, stepping
    over alternate samples to decimate by a factor of two (from Fs=5.65MHz,
    to 2.33MHz).  This sampling converter convolves coefficients calculated
    from the convolution of Goodman & Carey's filter 2 with itself 3 times.
    Data_1 array of arrays produced by filter 1 is used to refer to a look-
    up table of comb convolution outcomes, and the output is stored for use
    by filter 3 in the array of integer arrays, data_2.  nb: another filter
    may be used but the loop limits and array type should be checked!

    filter 3: A 15 tap, single precision, half-band LPFIR, with 4 different
    coefficients. Again, this decimates by a factor of two. It has symmetry
    in the coefficients enabling fewer multiplies to be operated, and comes
    pre-scaled to make the centre coefficient unity. Only half the filter's
    listed, starting with the adjacent-to-centre value,  and finishing with
    the last value (ie:only the second half of the filter). It's programmed
    in to reduce pointer calculations.

    filter 4: A 27 tap, single precision, half-band LPFIR, with 7 different
    coefficients. This is stored, operated and functions as filter 3 except
    for being a higher order and different frequency specifications.

    filter 5: A 163 tap half-band LPFIR, with convolution stepping over al-
    ternate samples, to effect decimation by a factor of two (to 352.8KHz).
    The filter is operated as filter 4, (with alternate zeros!), 'DEC2.BIN'
    is its coefficient file,  data_4 is its input array,  and 24 bit output
    formatted as Audio Precision's System One '.WAV' file is written to the
    filename supplied for the input (.RLC file), with the '.DEC' extension,
    via the 'output' array which is a memory copy. 'DEC2.BIN' is not scaled
    to have a unity central coefficient, to maintain downward compatability
    with previous versions of 'DECIMATE.?00' programs.

    The 256 byte header required is read in from binary file,'CH1-CH2.HED',
    and the coefficients are read from binary files of coefficients.

                                                    R.E.H. - 19/07/91,
                                          fully half-band :  22/10/91.
                              LONGINT casting added (cf.ver. 5.00) : 05/01/92. }


    { DECLARATIONS & DEFINITIONS }

USES DOS;

TYPE coefficient_array = ARRAY [0..63]    OF single   ; { 50 terms => 2 blk }
  single_data_buffer_2 = ARRAY [0..4095]  OF single   ; { filt 4 o/p:1/2 in }
  single data_buffer_1 = ARRAY [0..8191]  OF single   ; { filt 3 o/p:1/2 in }
   integer_data_buffer = ARRAY [0..16383] OF integer  ; { filt 2 o/p:1/2 in }
      byte_data_buffer = ARRAY [0..32767] OF byte     ; { comb o/p:16/input }
        AP_wave_format = ARRAY [0..49407] OF byte     ; { 16k*3+256 header  }
             rlc_array = ARRAY [0..18575] OF shortint ; { 18K for filter 1. }
```

```
VAR filter    : ^coefficient_array                    ;
    input     : ^rlc_array                            ;
    output    : ^AP_wave_format                       ;
    data_4    :  ARRAY[0..8] OF ^single_data_buffer_2 ; { nb: 'data' suffix }
    data_3    :  ARRAY[0..8] OF ^single_data_buffer_1 ; { is the No. of the }
    data_2    :  ARRAY[0..8] OF ^integer_data_buffer  ; { filter from which }
    data_1    :  ARRAY[0..8] OF ^byte_data_buffer     ; { it comes.         }
    comb      :  ARRAY[0..35] OF integer              ;

    iofile                                          : file    ;
    counter, temp_long                              : longint ;
    temp_int                                        : integer ;
    hour, min, sec, loopcount, index, temp_word     : word    ;
    temp_real                                       : real    ;


   { PROGRAM TITLE & INPUT SETUP }

BEGIN
   IF paramcount <> 1 THEN EXIT;
   IF MEMAVAIL < 470500 THEN
   BEGIN
      WRITE(CHR(27), '[2J', CHR(27), '[6;26H', 'Insufficient memory',
            CHR(27),'[20;0H');
      EXIT;
   END;
   WRITE(CHR(27),'[2J',CHR(27),'[1;32m',CHR(27),'[6;26H',
         'RLC to AP (ver 6.00)');

   FOR index := 0 TO 17 DO
   BEGIN                                            { defining the accumulated }
      comb[index]:=(index*(index+1)) SHR 1 -162;    { convolution for overlaps }
      comb[35-index] := - comb[index]               ; { from 0-35, this notation }
   END                                              ; { allows data_1 to be byte }

   FOR index := 0 TO 8 DO NEW (data_1[index])       ; { define data_1 first to }
   NEW (input)                                      ; { give contiguous data_2 }
   FILLCHAR(input^,18576,0)                         ; { after input's removed. }

   ASSIGN (iofile, paramstr(1)+'.rlc')              ; { <= 129 blocks, 18K poss }
   RESET (iofile)                                   ; { 2's comp. format.  note }
      BLOCKREAD(iofile,input^,FILESIZE(iofile))     ; { zeroed, (FILLCHAR), for }
   CLOSE(iofile)                                    ; { when input's undefined. }


   {  FILTER 1 : GAIN = 162 }

   GETTIME(hour,min,sec,temp_word);
   WRITE(CHR(27),'[34m', CHR(27),'[8;18H',
         'First filter entered at : ',hour,':',min,'.',sec, CHR(27),'[s');
   FOR index := 0 TO 8 DO                           { This 9x 2K output defines }
   BEGIN                                            { the rlc array size at 18K }
      FOR counter := 0 TO 32767 DO
      BEGIN
         data_1[index]^[counter] := 18;                          { 18 overlap }
         temp_word := input^[counter SHR 4 + index SHL 11]+144 ;
         loopcount := counter;
         WHILE temp_word >= 35 DO
         BEGIN
            INC(loopcount);
            data_1[index]^[loopcount]:= 35;                      { 35 overlap }
            DEC(temp_word,18);
         END;
         INC(loopcount);
         data_1[index]^[loopcount] := temp_word;            { MOD 35 overlap }
         IF temp_word <> 17 THEN
         BEGIN
            DEC(temp_word,18);
            INC(loopcount);
            data_1[index]^[loopcount] := temp_word;         { MOD 18 overlap }
         END;
         WHILE (loopcount-counter) < 15 DO
         BEGIN
            INC(loopcount);
            data_1[index]^[loopcount]:= 0 ;                       { 0 overlap }
         END;
         INC(counter,15);
      END;
   END;
   GETTIME(hour,min,sec,temp_word);
   WRITE(CHR(27),'[u,',CHR(27),'[9;29H',
         'completed at : ',hour,':',min,'.',sec,'.');
   DISPOSE(input);


   { FILTER 2 : GAIN = 64 }

   FOR index := 0 TO 8 DO                                   { coincident }
      data_2[index] := PTR( LONGINT(data_1[index]) SHR 16,     { segment }
                            LONGINT(data_1[index]) AND 65535 );  { offset }
   GETTIME(hour,min,sec,temp_word);
   WRITE(CHR(27),'[10;17H','Second filter entered at : ',
         hour,':',min,'.',sec, CHR(27),'[s');
   FOR counter := 0 TO 263571 DO    { m*(elements*arrays+overrun) from zero }
   BEGIN                            { 2*( 16384  *   8  +  714  )    -1     }
      index     := counter SHR 15;
      temp_word := counter AND 32767;
      temp_int  := comb[data_1[index]^[temp_word]]
                     +comb[data_1[index]^[temp_word+6]]
                +6*(comb[data_1[index]^[temp_word+1]]
                     +comb[data_1[index]^[temp_word+5]])
               +5*(4*comb[data_1[index]^[temp_word+3]]
                 +3*(comb[data_1[index]^[temp_word+2]]
                     +comb[data_1[index]^[temp_word+4]]));
      data_2[index]^[(temp_word) SHR 1] := temp_int;
      INC(counter);
   END;
   GETTIME(hour,min,sec,temp_word);
   WRITE(CHR(27),'[u,',CHR(27),'[11;29H',
         'completed at : ',hour,':',min,'.',sec,'.');
```

```
{ FILTER 3 : GAIN = 2 }

FOR index := 0 TO 8 DO                                          { coincident }
   data_3[index] := PTR( LONGINT(data_2[index]) SHR 16,          { segment }
                         LONGINT(data_2[index]) AND 65535 );     { offset }
GETTIME(hour,min,sec,temp_word);
WRITE(CHR(27),'[12;18H','Third filter entered at : ',
      hour,':',min,'.',sec, CHR(27),'[s');
FOR counter := 0 TO 131771 DO    { m*(elements*arrays+overrun) from zero }
BEGIN                            { 2*(  8192  *   8  +  350  )    -1     }
   index     := counter SHR 14;
   temp_word := counter AND 16383;
   temp_real := data_2[index]^[temp_word+7]
                      +0.60082174*LONGINT(data_2[index]^[temp_word+6]
                                         +data_2[index]^[temp_word+8])
                      -0.12454963*LONGINT(data_2[index]^[temp_word+4]
                                         +data_2[index]^[temp_word+10])
                     +0.026778438*LONGINT(data_2[index]^[temp_word+2]
                                         +data_2[index]^[temp_word+12])
                    -0.0030508398*LONGINT(data_2[index]^[temp_word]
                                         +data_2[index]^[temp_word+14]);
   data_3[index]^[(temp_word) SHR 1]:= temp_real;
   INC(counter);
END;
GETTIME(hour,min,sec,temp_word);
WRITE(CHR(27),'[u,',CHR(27),'[13;29H',
      'completed at : ',hour,':',min,'.',sec,'.');

{ FILTER 4 : GAIN = 2 }

FOR index := 0 TO 8 DO NEW (data_4[index]) ;
GETTIME(hour,min,sec,temp_word);
WRITE(CHR(27),'[14;17H','Fourth filter entered at : ',
      hour,':',min,'.',sec, CHR(27),'[s');
FOR counter := 0 TO 65859 DO     { m*(elements*arrays+overrun) from zero }
BEGIN                            { 2*(  4096  *   8  +  162  )    -1     }
   index     := counter SHR 13;
   temp_word := counter AND 8191 ;
   temp_real := data_3[index]^[temp_word+13]
                      +0.62124836*(data_3[index]^[temp_word+12]
                                  +data_3[index]^[temp_word+14])
                      -0.169985728*(data_3[index]^[temp_word+10]
                                  +data_3[index]^[temp_word+16])
                      +0.068000484*(data_3[index]^[temp_word+8]
                                  +data_3[index]^[temp_word+18])
                      -0.025669274*(data_3[index]^[temp_word+6]
                                  +data_3[index]^[temp_word+20])
                     +0.0079741072*(data_3[index]^[temp_word+4]
                                  +data_3[index]^[temp_word+22])
                     -0.0017850053*(data_3[index]^[temp_word+2]
                                  +data_3[index]^[temp_word+24])
                    +0.00021796504*(data_3[index]^[temp_word]
                                  +data_3[index]^[temp_word+26]);
   data_4[index]^[(temp_word) SHR 1] := temp_real;
   INC(counter);
END;
GETTIME(hour,min,sec,temp_word);
WRITE(CHR(27),'[u,',CHR(27),'[15;29H','completed at : ',
      hour,':',min,'.',sec,'.');
FOR index := 0 TO 8 DO DISPOSE (data_1[index]) ;

{ FILTER 5 : GAIN = 1 }

NEW (output);
NEW (filter);
ASSIGN (iofile, 'dec2.bin')      ; { half-band => interleaved zeros used }
RESET (iofile)                   ;
   BLOCKREAD (iofile, filter^,2) ; { 41 non-trivial LPFIR-3 coefficients }
CLOSE(iofile)                    ;       { DC gain = 1.0 : sum(coeffs).  }
GETTIME(hour,min,sec,temp_word);
WRITE(CHR(27),'[16;18H','Fifth filter entered at : ',
      hour,':',min,'.',sec,CHR(27),'[s');
FOR counter := 0 TO 32767 DO                        { 4096*8 and MOD 2 = 1 }
BEGIN
   index     := counter SHR 12;
   temp_word := counter AND 4095 + 82;
   temp_real := data_4[index]^[temp_word] 2;      { centre coeff = 0.500 }
   loopcount := 1;
   REPEAT
      temp_real := temp_real + filter^[loopcount SHR 1] *
                               (data_4[index]^[temp_word+loopcount]
                               +data_4[index]^[temp_word-loopcount]);
      INC(loopcount,2);              { <-- alternate zeros skipped here }
   UNTIL loopcount = 83;
   temp_long := ROUND(202*temp_real);      { 2^23,/41472 ... gain comp.) }
   temp_word := 256+counter+counter SHR 1;
   output^[temp_word]:= (temp_long SHR 16) AND 255;
   INC(temp_word);
   output^[temp_word]:= (temp_long SHR 8) AND 255;
   INC(temp_word);
   output^[temp_word]:= temp_long AND 255;
   INC(counter);
END;
GETTIME(hour,min,sec,temp_word);
WRITE(CHR(27),'[u,',CHR(27),'[17;29H','completed at : ',
      hour,':',min,'.',sec,'.');

{ OUPUT & TIDY UP }

WRITE(CHR(27),'[19;22H',CHR(27),'[0;1;5m',
      CHR(27),'[sFormatting output file... ') ;
ASSIGN (iofile, 'ch1-ch2.hed')   ;
RESET (iofile)                   ;
   BLOCKREAD (iofile, output^,2) ;                { 256 byte ch1 header }
CLOSE(iofile)                    ;
ASSIGN (iofile, paramstr(1)+'.dec') ;
REWRITE(iofile)                  ;
   BLOCKWRITE(iofile,output^,386)   ;
CLOSE(iofile)                    ;
RELEASE (heaporg) ;
WRITE(CHR(27),'[u',CHR(27),'[0;1;31m',
      'Output''s stored in ',paramstr(1)+'.dec ',
      CHR(27),'[0m',CHR(27),'[5B',CHR(7));
END.
```

A.2.2 : IIR Decimator.

Decimation by any factor can be compiled into the following programme, 'IIR-DEC.PAS', which currently uses IIR filtering to perform 16x sample rate reduction. This programme uses compact (integer) input and produces '.wav' format output (renamed '.id8') and required an ASCII coefficient file, 'IIR-DEC.ASC' and an Audio Precision header file, 'IIR-DEC.HED', in the current directory.

IIR-DEC.PAS

```
PROGRAM to_decimate_by_a_factor_of_16_using_iir_filters;
{$N+}

TYPE heaparray = ARRAY [0..16383] OF integer ;
     AP_format = ARRAY [0..49407] OF byte    ;

VAR   file_in, file_out              : FILE                        ;
      coeff_file                     : TEXT                        ;
      index, offset, n, percent_done : WORD                        ;
      temp_long, first_val, counter  : LONGINT                     ;
      K, B, C, D                     : ARRAY [1..4] OF REAL        ;
      IIR_data                       : ARRAY[1..5,1..3] OF REAL    ;
      data                           : ARRAY [0..8] OF ^heaparray  ;
      wav                            : ^AP_format                  ;

BEGIN
   IF PARAMCOUNT <> 1 THEN
   BEGIN
       WRITELN('usage: filename (?.int)')  ;
       HALT                                ;
   END                                     ;

   IF MEMAVAIL < 344320 THEN
   BEGIN
       WRITELN ('insufficient memory') ;
       HALT                            ;
   END                                 ;

   WRITE(CHR(13),'reading data...')    ;

   FOR index := 0 TO 8 DO NEW(data[index]) ;
   ASSIGN (file_in, PARAMSTR(1)+'.int')    ;
   RESET (file_in)                         ;
   FOR index := 0 TO 7 DO BLOCKREAD(file_in, data[index]^,256) ;
   BLOCKREAD(file_in, data[8]^,12)         ;
   CLOSE(file_in)                          ;

   NEW(wav)                                ;
   ASSIGN (file_in, 'iir-dec.hed')         ;
   RESET(file_in)                          ;
   BLOCKREAD(file_in, wav^, 2)             ;
   CLOSE(file_in)                          ;

   ASSIGN (coeff_file,'iir-dec.asc');
   RESET (coeff_file)              ;
   FOR n := 1 TO 4 DO
   BEGIN
       READLN(coeff_file, K[n])    ;
       READLN(coeff_file, B[n])    ;
       READLN(coeff_file, C[n])    ;
       READLN(coeff_file, D[n])    ;
   END                             ;
   CLOSE(coeff_file)               ;

   WRITE(CHR(13),CHR(27),'[5minitialising...') ;

   FOR n := 1 TO 5 DO
   BEGIN
      IIR_data[n,1] := 0.0 ;
      IIR_data[n,2] := 0.0 ;
      IIR_data[n,3] := 0.0 ;
   END                     ;

   first_val := data[0]^[0] ;
   percent_done := 0        ;

{  This decimates by 16, using an 8th order IIR, configured as a series of
   bi-quad sections. It uses coefficients from the arrays K,B,C,D. Interim
   answers are stored in 'IIR_data'.                                       }

   FOR counter := 0 TO 131839 DO
   BEGIN
       IF (100*(counter/131840) > percent_done)
         AND (percent_done <> 0)
           THEN
             BEGIN
               IF (percent_done=1) THEN
                   WRITE(CHR(13),CHR(27),'[0m',CHR(27),'[K') ;
               INC(percent_done)                             ;
               WRITE(CHR(13),percent_done-1:2,'%')           ;
             END                                             ;

       index  := counter SHR 14    ;
       offset := counter AND 16383 ;

       IIR_data[1,1] := data[index]^[offset] ;

       IF (IIR_data[1,1]-first_val < 10)       { smart initialisation }
         AND (percent_done = 0)
           THEN
             BEGIN
               counter := 0      ;
               INC(percent_done) ;
             END                 ;
```

```
    FOR n := 1 TO 4 DO                                    { four biquads }
    BEGIN
        IIR_data[n,1]   := K[n]* IIR_data[n,1]        { recursive part }
                         - C[n]* IIR_data[n,2]
                         - D[n]* IIR_data[n,3] ;
        IIR_data[n+1,1] := IIR_data[n,1]              { non-recursive part }
                         + B[n]* IIR_data[n,2]
                         + IIR_data[n,3]       ;
        IIR_data[n,3] := IIR_data[n,2]           ;            { shift }
        IIR_data[n,2] := IIR_data[n,1]           ;            { shift }
    END                                          ;

    IF ((counter - 48) MOD 16 = 0)              { decimating by 16 }
      AND (counter >= 48)                      { skip initial output, }
        AND (percent_done <> 0)               { not initialising filters }
          THEN
            BEGIN
              temp_long := ROUND(256.*IIR_data[5,1])  ;
              IF (ABS(temp_long) > 8388607) THEN
                  WRITE(' last overflow @',counter)  ;
              offset := (counter - 48) SHR 4         ;   { M=16 }
              wav^[258+3*offset] := temp_long AND 255 ;
                    temp_long := temp_long SHR 8     ;
              wav^[257+3*offset] := temp_long AND 255 ;
                    temp_long := temp_long SHR 8     ;
              wav^[256+3*offset] := temp_long AND 255 ;
            END                                      ;
  END ;
  FILLCHAR(wav^[12544],36864,0) ;           { zero pad to 16384 samples }
  ASSIGN (file_out, PARAMSTR(1)+'.idF') ;
  REWRITE(file_out)                     ;
  BLOCKWRITE(file_out, wav^, 386)       ;
  CLOSE(file_out)                       ;

  RELEASE(heaporg) ;
  WRITE(CHR(7))    ;
END.
```

A.2.3 : Sideband estimation programme

In the following programme, the theoretical level of sideband from various modulation types (as analysed in appendix A8) is used to iteratively find the level to which PWM is noise sensitive at the input. 'FOLDBACK.PAS' uses a geometric version of a binary search to find the tonal level in successive bands which would corrupt a specified level within the audio band with particular signal bandwidth and sample rate. This process can be applied for each carrier harmonic in turn, with the results written to a '.dat' format file for viewing using Audio Precision's 'System One' software.

FOLDBACK.PAS

```
PROGRAM to_find_the_intermod_amplitude_of_PWM_which_spoils_the_audio_floor;
{$N+}
USES BESSEL;

FUNCTION dB2LIN(x:double):double;
BEGIN
   dB2LIN := EXP(x*Ln(10.0)/20.0);
END;
FUNCTION LIN2dB(x:double):double;
BEGIN
   LIN2dB := 20.0*Ln(x)/Ln(10.0) ;
END;

VAR sampling, modulation                    : CHAR    ;
    m, n, index, channel2                   : INTEGER ;
    Fc, Wc, Fv, Wv, Fb, Wb, noise_floor     : REAL    ;
    x, intermod, UDS_modifier, temp_double,
    min_amplitude, max_amplitude, amplitude : DOUBLE  ;

PROCEDURE GET_PARAMETERS;
BEGIN
    READLN(sampling);  { sampling type : uniform or natural; AD only! }
    READLN(modulation);     { PWM: trailing, double sided or 2 sample }
    READLN(Fc);                                  { sampling frequency }
    Wc := 2.0*pi*Fc;
    READLN(Fb);                                      { baseband width }
    Wb := 2.0*pi*Fb;
    READLN(noise_floor);                              { in-band floor }
    noise_floor := dB2LIN(noise_floor);
    READLN(m);                                { carrier harmonic to test }
    n := 2;                                        { starting harmonic }
    channel2 := 0;                                 { filler for output }
END;

PROCEDURE EVALUATE_INTERMOD;
BEGIN
    UDS_modifier  := 1.0;       { unless altered later }
    max_amplitude := 1.0;       { full scale }
    amplitude     := 1.0e-5 ;   { attempted centre point : geometric av.}
    min_amplitude := 1.0e-10;   { -200dB floor limit }

    CASE sampling OF
       'U':CASE modulation OF
             'T':temp_double := pi*(m*Wc - n*Wv)/Wc                ; { UTE }
             'D':BEGIN
                   temp_double :=(pi 2.0)*(m*Wc-n*Wv)/Wc           ; { UDS }
                   UDS_modifier:=SIN((pi 2.0)*(m*Wc-n*Wc+n*Wv)/Wc);
                 END;
             '2':temp_double := (pi/2.0)*(m*Wc - n*Wv)/Wc          ; { U2S }
           END;
```

```
        'N':CASE modulation OF
              'T'       : temp_double := pi*m                          ; { NTE }
              'D':temp_double := (pi/2.0)*m                            ; { NDS }
              '2':BEGIN
                    WRITELN('two sample''s a uniform modulation type !');
                    HALT;
                  END;
            ELSE BEGIN
                    WRITELN('invalid PWM type');
                    HALT;
                 END;
            END;
            ELSE BEGIN
                    WRITELN('invalid sampling type');
                    HALT;
                 END;
    END;
    REPEAT
        x := amplitude*temp_double;
        intermod := ABS(J(n,x)*UDS_modifier/temp_double) ;
        IF intermod < 1.0e-10 THEN intermod := 1.0e-10 ;
        IF intermod < noise_floor THEN min_amplitude := amplitude
                                  ELSE max_amplitude := amplitude ;
        amplitude := SQRT(max_amplitude*min_amplitude);     { geometric av.}
    UNTIL (ABS(1.0-intermod/noise_floor) < 0.001152)    { 0.01dB deviation }
       OR (min_amplitude > 0.98855);                              { -0.1dBFS }
END;

BEGIN   { main }
    GET_PARAMETERS;
    WRITELN('          Hz                dBFS                OFF');
    FOR index := 699 DOWNTO 0 DO
    BEGIN
        Wv := Wc * (index/1400.0);
        Fv := Wv/(2*pi);
        IF Wv < Wb THEN amplitude := noise_floor
                   ELSE IF Wv < (m*Wc-Wb)/n THEN
                   BEGIN
                       IF amplitude < -0.1 THEN INC(n);
                       amplitude := 1.0;
                   END
                   ELSE IF Wv >= m*Wc/n THEN amplitude := 1.0
                                        ELSE EVALUATE_INTERMOD;
        amplitude := LIN2dB(amplitude);
        WRITELN(Fv:17:0,',',amplitude:21:6,',',channel2:20);
    END;  { FOR }
END.
```

A.2.4 : Spectral SNR assessment

'SNR.PAS' can be used to read pre-stored spectrum files (parameter 1) as produced by Audio Precision's 'System One' software, and calculate the total harmonic distortion and noise ratio to the signal power from the frequency domain. This can be used for both hardware and software measurements by importing spectral results into the test set and storing '.dat' files for use with this programme. Signals within 200 Hz of the used supplied signal frequency (parameter 2) are considered a part of the signal and DC is ignored.

SNR.PAS

```
PROGRAM to_read_a_signal_spectrum_and_calculate_snr;
{$N+}
VAR AP_dat_file      : TEXT    ;
    header, temp_str : STRING  ;
    code, counter    : INTEGER ;
    frequency        : REAL    ;
    signal, noise    : DOUBLE  ;

BEGIN
    IF PARAMCOUNT <> 2 THEN EXIT;
    ASSIGN(AP_dat_file,PARAMSTR(1)+'.dat');
       RESET(AP_dat_file);
    VAL(PARAMSTR(2),frequency,code);
    WRITELN('signal frequency : ',frequency:8:3);
    READLN(AP_dat_file,header);
    READLN(AP_dat_file,header);  { skip first two data points & header }
    READLN(AP_dat_file,header);
    noise := 0;
    counter := 0;
    REPEAT
       READLN(AP_dat_file,header);
       temp_str := COPY(header,1,17);
       VAL(temp_str,signal,code);
       IF ABS(signal-frequency) > 200 THEN
       BEGIN
           temp_str := COPY(header,19,21);
           VAL(temp_str,signal,code);
           IF signal > -200.0 THEN
           BEGIN
              noise := noise + 2.0*EXP(signal*Ln(10.0)/10.0);
              INC(counter);
           END;
       END;
    UNTIL(EOF(AP_dat_file));
    noise := 10.0*Ln(noise)/Ln(10.0) ;
    WRITELN('noise power :  ',noise:15:8);
    CLOSE(AP_dat_file);
END.
```

A.2.5 : Spectral weight generator

For spectral SNR measurements in which a psychoacoustic weighting has been applied, spectral equalisation files are required which can be read by Audio Precision's "System One" software, and subtracted in frequency (bin by bin) to produce a new spectrum for SNR assessment. 'E_WEIGHT.PAS' is an example of a programme to produce one such equalisation file and is listed below.

E-WEIGHT.PAS

```
PROGRAM to_generate_a_modified_e_weighting_curve
{$N+}
TYPE complex = RECORD
                 RE,IM : REAL ;
               END ;

PROCEDURE P2R(VAR x:complex) ;
VAR y : complex ;
BEGIN
    complex(y).RE := complex(x).RE * COS(complex(x).IM) ;
    complex(y).IM := complex(x).RE * SIN(complex(x).IM) ;
    x := y ;
END ;

PROCEDURE R2P(VAR x:complex);
VAR y : complex ;
BEGIN
    complex(y).RE := SQRT(SQR(complex(x).RE)+SQR(complex(x).IM)) ;
    complex(y).IM := ARCTAN(complex(x).IM/(complex(x).RE+1E-20)) ;
    x := y ;
END ;

PROCEDURE CADD(VAR x,y,z:complex) ;
BEGIN
    complex(z).RE := complex(x).RE + complex(y).RE ;
    complex(z).IM := complex(x).IM + complex(y).IM ;
END;

PROCEDURE CCONJ(VAR x:complex) ;
BEGIN
    complex(x).IM := -complex(x).IM ;
END ;

PROCEDURE CMUL(VAR x,y,z:complex) ;
BEGIN
    complex(z).RE := complex(x).RE * complex(y).RE
                   - complex(x).IM * complex(y).IM ;
    complex(z).IM := complex(x).RE * complex(y).IM
                   + complex(x).IM * complex(y).RE ;
END ;

PROCEDURE CDIV(VAR x,y,z:complex) ;
VAR m,a,b : complex ;
BEGIN
    complex(m).RE := 1.0 SQRT(SQR(complex(y).RE)+SQR(complex(y).IM)) ;
    complex(m).IM := 0.0 ;
    CMUL(x,m,a) ;
    CMUL(y,m,b) ;
    CCONJ(b)    ;
    CMUL(a,b,z) ;
END ;

FUNCTION LIN2dB(x:REAL):REAL ;
BEGIN
    LIN2dB := 20.0*Ln(x)/Ln(10.0) ;
END ;

VAR
    index                  : INTEGER              ;
    frequency, channel2, w : REAL                 ;
    amplitude, x, y, z, s  : complex              ;
    K                      : ARRAY [1..7] OF REAL ;

BEGIN
    frequency := 22000.0 ; { ** remove for file interpretation ** }
    channel2 := 0.0      ;     { ** remove later if unused ** }
    K[1] := 600.0   ; { A }
    K[2] := 1000.0  ; { B }
    K[3] := 180.0   ; { C }
    K[4] := 1500.0  ; { D }
    K[5] := 900.0   ; { E }
    K[6] := 3600.0  ; { F }
    K[7] := 9.53E+8 ; { G }
    FOR index := 1 TO 6 DO K[index] := K[index] * -2.0 * PI ;

    WRITELN('              Hz                  dBFS                OFF') ;
    FOR index := 699 DOWNTO 0 DO
    BEGIN
        frequency := frequency * 0.99 ; { 0.99^700*22050 = 19.4 Hz }
        w := 2.0*PI*frequency ;
{ evaluate numerator }
        complex(s).RE := 0.0 ;
        complex(s).IM := w ;
        complex(x).RE := K[1] ;
        complex(x).IM := K[2] ;
        y := x ;
        CCONJ(y)    ; { old y lost ! }
        CADD(x,s,z) ;
        CADD(y,s,x) ;
        CMUL(x,z,y) ; { ans in y }
        CMUL(s,s,z) ; { s^2 }
        CMUL(z,s,x) ; { s^3 }
```

```
        CMUL(x,y,z) ; { ans in z }
        amplitude := z ;
{ evaluate denominator first term }
        complex(x).RE := K[3] ;
        complex(x).IM := 0.0 ;
        CADD(x,s,y) ;
        CMUL(y,y,x) ; { y^2 }
        CMUL(x,y,z) ; { y^3 }
        x := amplitude ;
        CDIV(x,z,amplitude) ;
{ evaluate denominator second term }
        complex(x).RE := K[4] ;
        complex(x).IM := 0.0 ;
        CADD(x,s,y) ;
        CMUL(y,y,z) ; { y^2 }
        x := amplitude ;
        CDIV(x,z,amplitude) ;
{ evaluate denominator 3rd & 4th terms }
        complex(x).RE := K[5] ;
        complex(x).IM := K[6] ;
        z := x ;
        CCONJ(z)    ; { old z lost ! }
        CADD(x,s,y) ;
        CADD(z,s,x) ;
        CMUL(x,y,z) ;
        x := amplitude ;
        CDIV(x,z,amplitude) ;
        R2P(amplitude) ; { old amplitude lost ! }
        w := K[7] * complex(amplitude).RE ; { temporary use of w }
        w := LIN2dB(w) ;
        WRITELN(frequency:17:0,',',w:21:6,',',channel2:20:8) ;
    END;
END.
```

Example output from this programme is plotted below (figure A.2.5.a) for reference.

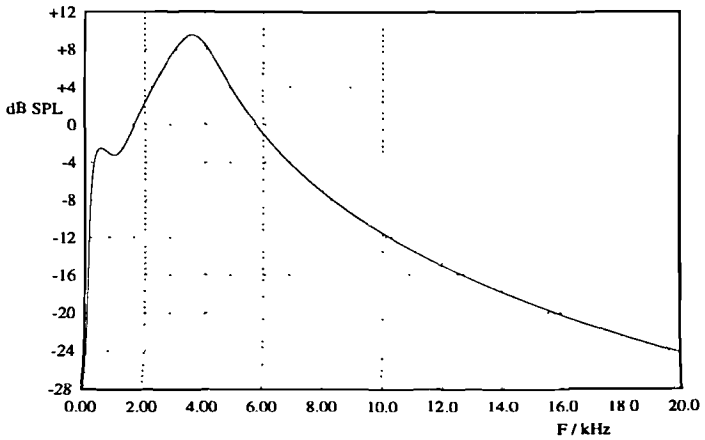


Figure A.2.5.a : E-Weighting Curve as used in Psychoacoustic Measurement and Design.

Appendix 3 : Filter design & optimisation software.

A.3.1 : Equi-ripple FIR filter order estimation

Programme 'ORDER.PAS' implements the empirically based equations presented in [GOO77] for estimating the order of filter required for a low pass, equi-ripple FIR filter with specified passband and stopband ripple and edge frequencies. Five parameters are taken from the command line for the order to be computed, and the answer provided can then be used in common design packages such as that written by Parks, McClellan & Rabiner, based on Remez exchange [MCC73b].

ORDER.PAS

```
PROGRAM  to_estimate_low_pass_linear_phase_equiripple_FIR_filter_order ;

{ revise compiler directives for debugging }
{$A+,B-,D-,E-,F-,I+,L-,N+,O-,R-,S-,V+}
{$M 16384,0,655360!}

{ variables : as defined p.149, Crochiere & Rabiner, "Multi-Rate DSP".
          N = order      (integer > 1)
         Ws = omega s    (the stopband cutoff frequency in radians)
         Wp = omega p    (the passband cutoff frequency in radians)
         ds = delta s    (the stopband ripple in linear measurement)
         dp = delta p    (the passband ripple in linear measurement)
          f = f(dp,ds)   (intermediate term 1)
          d = D(dp,ds)   (intermediate term 2; nb. d <> 'D' from the ref.    }

    VAR   code, error             : integer            ;
          f, d, N, Ws, Wp, ds, dp : real               ;
          a                       : ARRAY[1..6] OF real ;

   BEGIN

{ define the empirical constants }
      a[1] := 0.005309 ;
      a[2] := 0.071140 ;
      a[3] := -0.47610 ;
      a[4] := -0.00266 ;
      a[5] := -0.59410 ;
      a[6] := -0.42780 ;

{ check the command line }
      IF PARAMCOUNT <> 5 THEN
      BEGIN
         WRITELN('usage: Fs, fp, fs, dp, ds (freq./Hz, ripple/dB, all +ve)');
         HALT                                                               ;
      END                                                                   ;

{ interpret the command line : nb. reverse order read for subsequent write }
      error := 0                        ;
      VAL(PARAMSTR(5),ds,code)          ; { temporarily quoted in dBs }
      IF code <> 0 THEN INC(error) ;
      IF ((ds<0.0) OR (ds>200.0)) THEN error := error OR 1 ; { range check }
      error := error SHL 1              ;
      VAL(PARAMSTR(4),dp,code)          ; { temporarily quoted in dBs }
      IF code <> 0 THEN INC(error) ;
      IF ((dp<0.0) OR (dp>200.0)) THEN error := error OR 1 ; { range check }
      error := error SHL 1              ;
      VAL(PARAMSTR(3),Ws,code)          ; { borrow 'Ws' for 'fs' temporarily }
      IF code <> 0 THEN INC(error) ;
      IF Ws < 0.0 THEN error := error OR 1 ; { polarity check }
      error := error SHL 1              ;
      VAL(PARAMSTR(2),Wp,code)          ; { borrow 'Wp' for 'fp' temporarily }
      IF code <> 0 THEN INC(error) ;
      IF Wp < 0.0 THEN error := error OR 1 ; { polarity check }
      IF Wp > Ws THEN error := error OR 3 ; { check *LPF* specification }
      error := error SHL 1              ;
      VAL(PARAMSTR(1),f,code)           ; { borrow 'f' for 'Fs' temporarily }
      IF code <> 0 THEN INC(error) ;
      IF f < 0.0 THEN error := error OR 1 ; { polarity check }
      IF f < Ws THEN error := error OR 1 ; { check appropriate Fs wrt Ws }

      IF error <> 0 THEN
      BEGIN
         FOR code := 1 to 5 DO
         BEGIN
            IF (error AND 1) <> 0 THEN
               WRITELN('parameter ', code:1, ' is incorrectly entered.') ;
            error := error SHR 1 ;
         END  ;
         HALT ;
      END ;

{ normalise the frequencies in radians }
      Wp := 2.0*PI*Wp f ;
      Ws := 2.0*PI*Ws f ;

{ convert the ripple figures from dBs to linear ... inefficient but clearer }
      dp := EXP(dp*Ln(10.0) (20.0))-1.0 ;
      ds := EXP(ds*Ln(0.1)/(20.0))      ; { .1 used since stopband dBFS -ve }

{ check the ripple parameter suitability }
      IF dp < ds THEN WRITELN('the passband''s tighter than the stopband; ',
                              'the solution may be inaccurate.') ;
```

```
{ calculate intermediate terms }
      n := Ln(dp)/Ln(10.0)          ; { temporary use of 'n' for Log10(dp). }
      f := Ln(ds)/Ln(10.0)          ; { Fs overwritten, f(dp,ds) incomplete }
      d := f*((a[1]*n+a[2])*n+a[3]) ; { D(dp,ds) first term,  }
      d := d+(a[4]*n+a[5])*n+a[6]   ; { D(dp,ds) complete...  }
      f := 11.012-0.512*(n-f)       ; { f(dp,ds) completed.   }
      N := 2.0*PI                   ; { Log10(dp) overwritten }
      N := (Ws-Wp)/N                ; { N becomes the transition bandwidth. }
      N := d/N-f*N+1.0              ; { transition width lost, N completed. }

{ check calculation validity }
      IF ((d<0.0) OR (f<0.0) OR (N<0.0)) THEN     { nb. type compatability! }
         WRITELN('there appear to be errors in the intermediate terms;',
                 '                             the solution may be inaccurate.') ;

{ 'round up' to next highest integer and write out }
      N := TRUNC(N)+1.0             ;
      WRITELN('Recommended linear phase equiripple filter order is', N:4:0) ;

      WRITE(CHR(7))     ;
   END.
```

A.3.2 : Linear to minimum phase filter conversion

Two techniques exist for linear to minimum phase conversion of a sequence; firstly: deflation of the sequence as a polynomial, followed by inversion of roots with radius greater then unity and re-expansion of the new roots. This becomes prone to error for long sequences since the deflation of a polynomial can become ill-conditioned, but is useful for short sequences (such as the impulse responses of noise transfer functions). Programme 'LAGUERRE.FOR' is listed below, and implements the deflation of a polynomial (the first part of conversion to minimum phase). The output from this can be used to display the roots for display by standard routines such as the UNIX utility 'graph' or the X11 utility 'xgraph' or tabulated for subsequent inversion and expansion.

LAGUERRE.FOR

```
      PROGRAM TO FIND THE ROOTS OF AN FIR
c
c nb: REAL roots cannot be guaranteed, so techniques such as Muller's
c method cannot be applied, and a general (slower) technique which is
c guaranteed to converge on real, complex, single and multiple roots
c has to be used.
c Laguerre's method is used here instead of the Jenkins-Traub method
c (ref: Ralston & Rabinowitz) or the Lehmer-Schur algorithm (single
c dimensional bracketing, ref: Acton), since is is simpler.
c
      REAL MAXVAL
      LOGICAL GO4IT
      INTEGER ORDER,I
      DIMENSION C(100)
      COMPLEX A(100),ROOTS(100)
c
c read in the filter order & coefficients
c
      READ(5,*) ORDER
      DO 1 I=ORDER+1,1,-1
           READ(5,*) C(I)
           A(I)=CMPLX(C(I),0.)
1     CONTINUE
c
c check if high accuracy is required
c
      IF (ORDER.GE.5) THEN
          GO4IT=TRUE
      ENDIF
c
c find the roots
c
      CALL ZROOTS(A,ORDER,ROOTS,GO4IT)
c
c find the maximum rel or imaginary value
c
      MAXVAL=1.0              ! set minimum as the unit circle
      DO 2 I=1,ORDER
           IF  (ABS(REAL(ROOTS(I))).GT.MAXVAL)
     *                              MAXVAL=  ABS(REAL(ROOTS(I)))
           IF (ABS(AIMAG(ROOTS(I))).GT.MAXVAL)
     *                              MAXVAL= ABS(AIMAG(ROOTS(I)))
2     CONTINUE
      MAXVAL=1.1*MAXVAL       ! gives some clearance at the graph edge
c
c write out the xgraph header
c
      CALL HEADER(MAXVAL)     ! set up axes, unit circle & scale
c
c write out the answers for xgraph ... Re & Im(ROOTS(1..ORDER))
c
      WRITE(6,*) 'PixelMarkers: false'
      WRITE(6,*) 'StyleMarkers: true'
      DO 4 I=1,ORDER
           WRITE(6,3) REAL(ROOTS(I)), AIMAG(ROOTS(I))
3          FORMAT(2F15.9)
4     CONTINUE
      WRITE(6,*) '"zeros"'
      STOP
      END
```

```
      SUBROUT'NE HEADER(MAXVAL)
c
c write out the xgraph header
c
      REAL MAXVAL
      PARAMETER (PI=3.141592654)
      WRITE(6,*) 'TitleText: Complex Z-plane'
      WRITE(6,*) 'XUnitText: Real'
      WRITE(6,*) 'YUnitText: Imag.'
      WRITE(6,*) 'XLowLimit:  ', -MAXVAL
      WRITE(6,*) 'XHighLimit: ',  MAXVAL
      WRITE(6,*) 'YLowLimit:  ', -1.162*MAXVAL   ! ...allow for default
      WRITE(6,*) 'YHighLimit: ',  1.162*MAXVAL   ! display aspect ratio
      WRITE(6,*) 'NoLines: true'
      WRITE(6,*) 'PixelMarkers: true'
      WRITE(6,*) 'StyleMarkers: false'
      DO 2 I 0,999
           WRITE(6,1) SIN(I*2.*PI/1000.),COS(I*2.*PI/1000.)
1          FORMAT(2F15.9)
2     CONTINUE
      WRITE(6,*) '"unit circle"'
      WRITE(6,*)
      RETURN
      END

      SUBROUTINE ZROOTS(A,M,ROOTS,POLISH)
c
c     (ref: p265, Numerical Recipes)
c
c Given degree M, M+1 complex coefficients A, this sucessively calls
c LAGUER with initial guess X (0,0), to find ROOTS of the polynomial
c relying on f rward deflation for guaranteed convergence.
c
c If root polishing is required, Laguerre can be called ignoring the
c error limit and looking to achieve accuracy to within the roundoff
c limit of the routine. See the subroutine's header for details.
c
      PARAMETER (EPS 1.E-6,MAXM 101) ! initial accuracy & max. order
      COMPLEX A(M+1),ROOTS(M),AD(MAXM),X,B,C
      LOGICAL POLISH
      DO 1 J 1,M+1              ! copy coeffs for deflation later
           AD(J) A(J)
1     CONTINUE
      DO 3 J M,1, 1             ! loop over each root to be found
           X CMPLX( .,0.)       ! zer  fav urs smallest remaining root
           CALL LAGUER(AD,J,X,EPS,.FALSE.)
           IF (ABS(AIMAG(X)).LE.2.*EPS**2*ABS(REAL(X))) THEN
              X CMPLX(REAL(X),0.)
           ENDIF
           ROOTS(J) X
           B AD(J+1)
           DO 2 JJ=J,1,-1
                C AD(JJ)
                AD(JJ) B
                B X*B+C
2          C NTINUE
3     C NTINUE
      IF (POLISH) THEN          ! polish roots - undeflated coeffs
          DO 4 J 1,M
               ALL LAGUER(A,M,ROOTS(J),EPS,.TRUE.)
4         CONTINUE
      ENDIF
      DO 6 J 2,M
           X ROOTS(J)
           DO 5 I J 1,1, 1
                IF(REAL(ROOTS(I)).LE.REAL(X)) GO TO 10    ! YAOW
                ROOTS(I+1) ROOTS(I)
5          CONTINUE
           I 0
10         ROOTS(I+1) X
6     C NTINUE
      RETURN
      END

      SUBROUTINE LAGUER(A,M,X,EPS,POLISH)
c
c     (ref: p264, Numerical Recipes)
c
c from the degree, M, and M+1 complex coeffs, A, accuracy, EPS, and
c initial c mplex value, X, this routine improves X by Laguerre's
c method until it c nverges to a root of the polnomial made from A.
c With POLISH true, EPS is ignored, and the routine attempts to
c impr ve the r  t to the achievable roundoff limit so a good initial
c guess is required under these conditions. The stopping criterion is
c discussed in Adams, Communications of the A.C.M., 10, 655, [1967].
c
      COMPLEX A(M+1),X,DX,X1,B,D,F,G,H,SQ,GP,GM,G2,ZERO
      LOGICAL POLISH
      PARAMETER (ZERO (0.,0.),EPSS=6.E-8,MAXIT=100)
      DXOLD CABS(X)                     ! change EPSS at your peril !
      DO 2 ITER 1,MAXIT                 ! iterate MAXIT times
           B A(M+1)
           ERR CABS(B)
           D ZERO
           F ZERO
           ABX CABS(X)
           DO 1 J M,1,-1                ! compute polynomial
                F X*F+D                 ! & its 1st 2 derivatives
                D X*D+B
                B X*B+A(J)
                ERR CABS(B)+ABX*ERR
1          CONTINUE
           ERR EPSS*ERR                    ! estimate roundoff error
           IF (CABS(B).LE.ERR) THEN        ! root found
               DX ZERO
               RETURN
           ELSE                            ! use Laguerre's method
               G D B
               G2=G*G
               H G2 2.*F B
               SQ CSQRT((M-1)*(M*H-G2))
               GP G+SQ
               GM=G SQ
```

```
            IF (CABS(GP).LT.CABS(GM)) GP=GM
            DX=M GP
        ENDIF
        X1=X-DX
        IF (X.EQ.X1) RETURN                    ! converged
        X=X1
        CDX CABS(DX)
        IF (ITER.GT.6.AND.CDX.GE.DXOLD) RETURN  ! reached roundoff
        DXOLD CDX
        IF (.NOT.POLISH) THEN
            IF (CDX.LE.EPS*CABS(X)) RETURN      ! converged
        ENDIF
2     CONTINUE
      PAUSE 'too many iterations in Laguerre'
      RETURN
      END
```

The second technique for linear to minimum phase conversion is based on the discrete Fourier transform. Conversion can be achieved with two DFTs since the minimum phase part of a sequence will be represented in the the odd component of its complex cepstrum, and the real cepstrum can be approximated by a periodic cepstrum provided the sequence decays quickly. Programme 'MPC.FOR' is listed below, and implements this technique with scaling to force the first sequence value to unity. If supplied with a filter's impulse response as a sequence, the output will be the the impulse response of a minimum phase lag filter with the same frequency response but minimum gain for that shape.

MPC.FOR

```
      PROGRAM MINIMUM PHASE FILTER CONVERSION
      EXTERNAL FFT, R8TR, ORD1, ORD2
      COMMON /FT/ PII, P7, P7TWO, C22, S22, PI2
      DOUBLE PRECISION PII, P7, P7TWO, C22, S22, PI2, C(256)
c
c     constants for the RADIX 8 FFT
      PII = 4.0D+0*ATAN(1.0D+0)
      P7 = 1.0D+0/SQRT(2.0D+0)
      P7TWO = 2.0D+0*P7
      C22 = COS(PII/8.0D+0)
      S22 = SIN(PII/8.0D+0)
      PI2 = 2.0D+0*PII
c     constant for the FFT
      N=512
c
c     open I/O units to override STDIO as supplied in the command line
c     this is available under Microsoft Fortran, and not Sun Fortran
c     OPEN(UNIT=5,FILE=' ',STATUS='UNKNOWN')
c     OPEN(UNIT=6,FILE=' ',STATUS='UNKNOWN')
c
      READ(5,*)FS      ! unformatted read to avoid float/int confusion
      DO 1 I=1,1000
         READ(5,1000,END=2)C(I)
   1  CONTINUE
   2  NOP=I-1
c
      CALL MPC(NOP,C,N)
c
      WRITE(6,1000)FS
      DO 3 I=1,NOP
         WRITE(6,1000)C(I)
   3  CONTINUE
1000  FORMAT(E15.9)
c     CLOSE(UNIT=5)    ! not required unless the units have been opened
c     CLOSE(UNIT=6)    ! not required unless the units have been opened
      STOP
      END

      SUBROUTINE MPC(NOP,C,N)
      EXTERNAL FFT, R8TR, ORD1, ORD2
c
c This has limited accuracy due to using 512 point FFT which may alias
c because it's short, and may be inaccurate through trying to evaluate
c on the unit circle where zeros may coincide with the 'sectors' caused
c by evaluating the fft at discrete bins 360/512 degrees apart.
c
      DOUBLE PRECISION A(514), C(256)
c
c note: C's the coefficients, A's what's sent to be fft-ed. C must not
c exceed half of A when copied in because of time aliasing; furthermore
c in the process of fft and ifft, the impulse response may be longer
c than expected (zero at evaluation points past the length of the filter
c but not in-between) so extra spare length should be provided.
c If dft/idft is available, the length chosen to avoid time aliasing is
c recommended to be > 3~5 x the filter length, and the length should
c only contain factors available in the available radix list.
c
c . represents 'the magnitude of'
c + in the first character of a write statement suppresses line feed
c nb: the above are not available on the suns
c
      DO 1 I=1,NOP
         A(I)=C(I)
   1  CONTINUE
c
c     zero-pad the rest of the FFT's input
      DO 2 I=NOP+1,N+2
         A(I)=0.0
   2  CONTINUE
c
      CALL FFT(A,N)      ! ideally DFT
```

```
c     replace Re(H(f)) with |H(f)| ( ~ ½(Re[H(f)]²+Im[H(f)]²) )
      DO 3 I=1,N+1,2
         A(I)=DSQRT(A(I)*A(I)+A(I+1)*A(I+1))
c        replace zeros in |H(f)| with 10^-8 (cf. Reddy)
         IF (A(I).LT.1.0E-8) THEN
             A(I)=1.0E-8
             WRITE(*,*)'+REPLACING ZEROs IN |H(f)|'
         ENDIF
c        & put Ln|H(f)| into the first half of A(I)
         A((I+1)/2)=DLOG(A(I))
    3 CONTINUE
c
c     reflect the 1st Ω of |H(f)| into the 2nd Ω, & clear the rest of A(I).
      DO 4 I=2,N/2
         A(N-I+2)=A(I)
    4 CONTINUE
      A(513)=0.0
      A(514)=0.0
c
c     calculate periodic approx. to real cepstrum of |H(f)| BY IDFFT. Since
c     Z = Z* for this case, the IDFT = DFT/N
c
      CALL FFT(A,N)
c
c     move Re(h(t))/N to the 1st Ω of A(I) & scale by 1/N (A[1] done later),
c     and calculate approx. to complex cepstrum of min. phase filter.
c     U(f) = 1.0, { f: 0, N/2   }
c          = 2.0, { f: 1..N/2-1 }  nb: only NOP values required.
      DO 5 I=2,NOP
         A(I)=2.0*A(2*I-1)/DBLE(N)
    5 CONTINUE
c
c     calculate min. phase filter's h(t) from its complex cepstrum
      C(1)=DEXP(A(1)/DBLE(N))
      C(2)=A(2)*C(1)
      DO 7 I=3,NOP
         C(I)=A(I)*C(1)
         DO 6 J=2,I-1
            C(I)=C(I)+(DBLE(J-1)/DBLE(I-1))*A(J)*C(I-J+1)
    6    CONTINUE
    7 CONTINUE
c
c     rescale the coefficients so that C(1)=1
      DO 8 I=NOP,1,-1
         C(I)=C(I)/C(1)
    8 CONTINUE
      RETURN
      END
```

A.3.3 : Simplex Noise Transfer Function Optimisation

To generate noise transfer functions with minimum gain for only partly specified frequency domain, optimisation can be used. 'MBNSOPT.FOR' was written to perform this function, using the Simplex technique as proposed by Nelmer & Mead [NEL64]. This technique is only a local minimiser, so repeated calls to the simplex routine are used, with a ramping number of moves per call, and restart from the best coefficients previously found. Although global optima are not guaranteed, this routine provides good approximations to the global optima in considerably less time than other techniques, without the need for special information such as function derivatives. This programme requires a starting point for the coefficients (which can be nowhere near optimal) and a parameter file supplying order, a flag to indicate the meaning of the first line of the start point file, number of frequency domain points to consider in the assessment of the noise stopband and the various points themselves (tabulated with frequency, required suppression and emphasis).

MBNSOPT.FOR

```
      PROGRAM MULTI BAND NOISE SHAPER OPTIMISER
      EXTERNAL ERROR, FFT, R8TR, ORD1, ORD2
      COMMON  EF  NOP, NOS, C, N, NOB, DELTA, WEIGHT
      COMMON /FT  PII, P7, P7TWO, C22, S22, PI2
      INTEGER NOB(514)
      DOUBLE PRECISION PII, P7, P7TWO, C22, S22, PI2,
     *                 FAC(50), A(20), C(20),
     *                 DELTA(514), WEIGHT(514),
     *                 AA, BB, CC, ERRLIM, STEP, FS
c
c     constants for the RADIX 8 FFT...explicitly double precision
      PII = 4.0D+0*ATAN(1.0D+0)
      P7 = 1.0D+0 SQRT(2.0D+0)
      P7TWO = 2.0D+0*P7
      C22 = COS(PII 8.0D+0)
      S22 = SIN(PII 8.0D+0)
      PI2 = 2.0D+0*PII
c
c
c     variable list : NOP = the number of parameters to be optimised
c                     NOS = the number of specifications to approximate
c                     FAC = the current modification factor set
c                       J = the i o file header - 2: gain, 1: Fs, 0: none
c                       A = the current coefficient set
c                       C = the original coefficient set ( C(1)=1.0! )
```

```
c                    FS = the input filter's sampling frequency
c                    PG = the input filter's power gain
c                     N = the length of the FFTs used in analysis (RADIX 8)
c                  DATA = the current N discrete frequency responses' set
c                   NOB = the bin No frequency specification set (max NOS)
c                 DELTA = the linear amplitude specification set (max NOS)
c                WEIGHT = the emphasis set satisfying delta wrt power gain
c
c     define : the first output coefficient to be 1. This is NOT updated !
      A(1)= 1.
c     start with the best power gain ludicrously large to fail first test
      FAC(49)=1.797693134862315D+308       ! IEEE max value (pl3 MS Fort ref)
c
c     read the variable parameters from the standard input
      READ(5,*) NOP, NOS, N, J
      DO 1 I=1,NOS
         READ(5,*) NOB(I), DELTA(I), WEIGHT(I)
    1 CONTINUE
c
c     open unit 15 for i/o
    2 CONTINUE
      OPEN(UNIT=15,FILE='CURRENT',STATUS='UNKNOWN')
c
c     read in the sample rate & the starting coefficients as directed by 'J'
      IF (J.EQ.2) THEN
          READ(15,1003)PG
      ENDIF
      IF (J.EQ.1) THEN
          READ(15,1002)FS
      ENDIF
      DO 3 I=1,NOP+1
         READ(15,1003)C(I)
    3 CONTINUE
      CLOSE(UNIT=15)
c
c     check that best power gain is retained for reference despite input read
      IF (FAC(49).LT.PG) THEN
         PG=FAC(49)
      ENDIF
c
c     loop to monitor convergence
      DO 7 IT=0,100,5
c
c     start with the coefficient modification factors at unity
      IF (IT .EQ. 0) THEN
         NSTOP=0
         DO 4 I=1,NOP
            FAC(I)=1.0
    4    CONTINUE
      ENDIF
c
c     set up the simplex parameters (see also subroutine SIMPLEX )
      AA=1.0
      BB=0.5
      CC=1.5
      NSTOP=NSTOP+50
      STEP=1.05
      ERRLIM=1.0E-5
c
      CALL SIMPLEX(NOP,FAC,AA,BB,CC,STEP,ERRLIM,NSTOP,IER)
c
c     calculate the new estimates of the coefficients
      DO 5 I=1,NOP
         A(I+1)=FAC(I)*C(I+1)
    5 CONTINUE
c
      OPEN(UNIT=15,FILE='CURRENT',STATUS='UNKNOWN')
c
c     output the sampling frequency and new coefficients as directed by 'J'
      IF (J.EQ.2) THEN
          WRITE(15,1003)FAC(49)
      ENDIF
      IF (J.EQ.1) THEN
          WRITE(15,1002)FS
      ENDIF
      DO 6 I=1,NOP+1
         WRITE(15,1003)A(I)
    6 CONTINUE
      IF (J.EQ.0) THEN
          WRITE(16,*)'Stop - Program terminated.'
      ENDIF
      CLOSE(UNIT=15)
c     end of output block                  ! IER = 0 (converged),
c                                          !     = 1 (not going to converge),
      IF (IER-1) 8,8,7                     !     = 2 (not yet converged).
c
c     end of loop to monitor convergence
    7 CONTINUE
c
    8 WRITE(6,*)'      power gain measured at ',FAC(49)
      WRITE(6,*)'passband failure measured at ',FAC(50)
      IF (FAC(49)-PG) 9,9,11
    9 OPEN(UNIT=15,FILE='BESTYET',STATUS='UNKNOWN')
      IF (J.EQ.2) THEN
          WRITE(15,1003)FAC(49)
      ENDIF
      IF (J.EQ.1) THEN
          WRITE(15,1002)FS
      ENDIF
      DO 10 I=1,NOP+1
         WRITE(15,1003)A(I)
   10 CONTINUE
      IF (J.EQ.0) THEN
          WRITE(16,*)'Stop - Program terminated.'
      ENDIF
      CLOSE(UNIT=15)
   11 GOTO 2
c
1001  FORMAT(1I3,2D15.9)
1002  FORMAT(E15.4)
1003  FORMAT(E15.9)
      STOP
      END
```

```
      SUBROUTINE SIMPLEX(NOP,FAC,A,B,C,STEP,ERRLIM,NSTOP,IER)
      EXTERNAL ERROR, PFT, RSTR, ORD1, ORD2
      DOUBLE PRECISION FAC(50),H(51),G(51,50),PBAR(50),PSTST(50),
     *                 PSTSU(50),PSTAR(50),FACT(50),
     *                 A,B,C,STEP,ERRLIM,ERROR,
     *                 P1,P2,T,HSTAR,HSTST,HSTSU,HMIN,HMAX
      DO 1 I=1,NOP
    1 G(1,I)=FAC(I)
      NP1=NOP+1
      IROW=2
      DO 7 I=1,NOP
      DO 9 J=1,NOP
    9 G(IROW,J)=G(1,J)
      G(IROW,I)=G(IROW,I)*STEP
      IROW=IROW+1
    7 CONTINUE
      NIT=1
      DO 200 J=1,NP1
      DO 201 K=1,NOP
      FACT(K)=G(J,K)
  201 FAC(K)=G(J,K)
      H(J)=ERROR(FAC)
  200 CONTINUE
      T=0.
   45 DO 2 I=1,NOP
      P1=DABS(FAC(I)-FACT(I))
      P2=P1/DABS(FACT(I))
      IF(P2 .GT. T) THEN
      T=P2
      END IF
    2 CONTINUE
      IF((T .LT. ERRLIM) .AND. (NIT .GT. 1)) THEN
      IER=0
      RETURN
      END IF
      DO 3 I=1,NOP
      FACT(I)=FAC(I)
    3 CONTINUE
      IMAX=1
      IMIN=1
      DO 13 I=2,NP1
      IF((H(I)-H(IMAX)) .LE. 0.) THEN
      GO TO 15
      END IF
      IMAX=I
   15 IF((H(I)-H(IMIN)) .GE. 0.) THEN
      GO TO 13
      END IF
      IMIN=I
   13 CONTINUE
      HMAX H(IMAX)
      HMIN=H(IMIN)
      DO 17 I=1,NOP
      PBAR(I)=0.
   17 CONTINUE
      DO 18 I=1,NP1
      IF((I-IMAX) .EQ. 0) THEN
      GO TO 18
      END IF
      DO 20 J=1,NOP
   20 PBAR(J)=PBAR(J)+G(I,J)
   18 CONTINUE
      DO 602 J=1,NOP
  602 PBAR(J)=PBAR(J) NOP
      DO 21 I=1,NOP
   21 PSTAR(I)=A*(PBAR(I)-G(IMAX,I))+PBAR(I)
      DO 202 K=1,NOP
  202 FAC(K)=PSTAR(K)
      HSTAR=ERROR(FAC)
      IF ((HSTAR-HMIN) .GE. 0.) THEN
      GO TO 23
      END IF
      DO 24 I=1,NOP
   24 PSTST(I)=C*(PSTAR(I)-PBAR(I))+PBAR(I)
      DO 203 K=1,NOP
  203 FAC(K)=PSTST(K)
      HSTST=ERROR(FAC)
      IF((HSTST-HMIN) .GE. 0.) THEN
      GO TO 26
      END IF
      DO 27 I=1,NOP
   27 G(IMAX,I)=PSTST(I)
      H(IMAX)=HSTST
      NIT=NIT+1
      IF(NIT .GT. NSTOP) THEN
      IER=2
      RETURN
      END IF
      GO TO 45
   23 DO 28 I=1,NP1
      IF((I-IMAX) .EQ. 0) THEN
      GO TO 28
      END IF
      IF((HSTAR-H(I)) .LT. 0.) THEN
      GO TO 26
      END IF
   28 CONTINUE
      IF((HSTAR-HMAX) .GT. 0.) THEN
      GO TO 31
      END IF
      DO 32 I=1,NOP
   32 G(IMAX,I)=PSTAR(I)
      HMAX=HSTAR
   31 DO 33 I=1,NOP
   33 PSTSU(I)=B*G(IMAX,I)+(1.-B)*PBAR(I)
      DO 205 K=1,NOP
  205 FAC(K)=PSTSU(K)
      HSTSU=ERROR(FAC)
   63 IF((HSTSU-HMAX) .GT. 0.) THEN
      IER=1
      RETURN
      END IF
      DO 36 I=1,NOP
```

```
 36 G(IMAX,I)=PSTSU(I)
    H(IMAX)=HSTSU
    NIT=NIT+1
    IF(NIT .GT. NSTOP) THEN
    IER=2
    RETURN
    END IF
    GO TO 45
 26 DO 40 I=1,NOP
 40 G(IMAX,I)=PSTAR(I)
    H(IMAX)=HSTAR
    NIT=NIT+1
    IF(NIT .GT. NSTOP) THEN
    IER=2
    RETURN
    END IF
    GO TO 45
    END
```

A.3.4 : Simulated Annealing Noise Transfer Function Optimisation

For global optimisation of the noise transfer function, simulated annealing can be used in place of the simplex routine presented in the last section. This has the disadvantage that it is very slow to converge on the best solution, but is less prone to stopping in local minima than the simplex. To speed up the basic idea of an annealer, the error rate at any one temperature is continually assessed, and the amount of trials required at the next temperature are varied. This assists with speeding up the convergence when most local solutions are poor, and slowing down the annealer when it might otherwise get stuck in a local minima. In the example below, a single target specification for the entire noise stopband ripple is specified, otherwise the parameter and start file are as required for the simplex routine presented above.

SANSOPT.FOR

```
      PROGRAM NOISE SHAPING FILTER OPTIMISER ! USING SIMULATED ANNEALING
      IMPLICIT NONE                          ! NOT MULTI BAND
      EXTERNAL ERROR, FFT, R8TR, ORD1, ORD2, RAN2
      COMMON /EF/ NOP, NOS, C, N, DELTA, WEIGHT
      COMMON /FT/ PII, P7, P7TWO, C22, S22, PI2
      INTEGER NOP,NOS,N,I,J
      DOUBLE PRECISION PII, P7, P7TWO, C22, S22, PI2,
     *                 FAC(50), A(20), C(20), DELTA, WEIGHT,
     *                 T, SCALE(20), RT, ERROR, FAIL, FS, PG
c
c     constants for the RADIX 8 FFT...explicitly double precision
      PII = 4.0D+0*ATAN(1.0D+0)
      P7 = 1.0D+0/SQRT(2.0D+0)
      P7TWO = 2.0D+0*P7
      C22 = COS(PII/8.0D+0)
      S22 = SIN(PII/8.0D+0)
      PI2 = 2.0D+0*PII
c
c     variable list : NOP = the number of parameters to be optimised
c                     NOS = the number of specifications to approximate
c                     FAC = the current modification factor set
c                       J = the output file type [0: nothing, 1: Fs, 2: PG]
c     (in main & error) A = the current coefficient set
c (anneal & metropolis) A = the marchesi parameter to measure convergence
c                       C = the original coefficient set ( C(1)=1.0! )
c                      FS = the sampling frequency (the input file header)
c                       N = the length of the FFTs used in analysis
c                    DATA = the current N discrete frequency responses' set
c                   DELTA = the passband ripple size (linear)
c                  WEIGHT = the emphasis on satisfying delta over power gain
c                   SCALE = the set of move sizes in each parameter
c                      NA = the set of numbers of accepted moves at 'scale'
c                      RT = the temperature reduction after NK*NS annealings
c                       T = the simulated annealing overall temperature
c                      see also the Marchesi variables in subroutine anneal.
c
c nb. FAC(49) and FAC(50) are modified by the error calculating function
c     to the values of the power gain and the weighted mean squared error.
c
c     define : the first output coefficient to be 1. This is NOT updated !
      A(1)= 1.
c
c     read in the general optimisation parameters as in MBNSOPT...
c     typical values : NOP=5,NOS=30,N=512,J=2,DELTA=5.62E-3,WEIGHT=1.E+6
c     & set up the annealing parameters (see also subroutine ANNEAL)
c     typical values : 2 > T > 0.1, SCALE(I) ~ 0.001, 0.95 > RT > 0.85
c
      OPEN(UNIT=15,FILE='ampar',STATUS='OLD')
c
      READ(15,1001) NOP
      READ(15,1001) NOS
      READ(15,1001) N
      READ(15,1001) J
      READ(15,1003) DELTA
      READ(15,1003) WEIGHT
      READ(15,1003) T
      READ(15,1003) SCALE(1)
      READ(15,1003) RT
c
      CLOSE(UNIT=15)
c
```

```
      OPEN(UNIT=15,FILE='start',STATUS='OLD')
c
c     read in the sample rate & the starting coefficients as directed by 'J'
      IF (J.EQ.1) THEN
          READ(15,1002)FS
      ENDIF
      IF (J.EQ.2) THEN
          READ(15,1002)FG
      ENDIF
      DO 1 I=1,NOP+1
         READ(15,1003) C(I)
    1 CONTINUE
c
      CLOSE(UNIT=15)
c
c     start with the coefficient modification factors at unity
c     and the scale of parameter moves as read into scale(1).
c
      DO 2 I=1,NOP
         SCALE(I)=SCALE(1)
         FAC(I)=1.0
    2 CONTINUE
c
c     define a record file
c
      OPEN(UNIT=17,FILE='status',STATUS='UNKNOWN')
c
      CALL ANNEAL(NOP,FAC,T,SCALE,RT)
c
      CLOSE(UNIT=17)
c
c     calculate the new estimates of the coefficients
      DO 3 I=1,NOP
         A(I+1)=FAC(I)*C(I+1)
    3 CONTINUE
      FAIL=ERROR(FAC)                    ! dummy call to evaluate the error
c
c     open the output file on unit 16
      OPEN(UNIT=16,FILE='bestyet',STATUS='UNKNOWN')
c
c     output the sampling frequency and new coefficients as directed by 'J'
      IF (J.EQ.1) THEN
          WRITE(16,1002)FS
      ENDIF
      IF (J.EQ.2) THEN
          WRITE(16,1002)FAC(49)
      ENDIF
      DO 4 I=1,NOP+1
         WRITE(16,1003)A(I)
    4 CONTINUE
      IF (J.EQ.0) THEN
          WRITE(16,*)'Stop - Program terminated.'
      ENDIF
      CLOSE(UNIT=16)
c
      WRITE(6,*)'passband failure measured at ',FAC(50)
 1001 F RMAT(I5)
 1002 FORMAT(E15.4)
 1003 FORMAT(E15.9)
      STOP
      END

      SUBROUTINE ANNEAL(NOP,FAC,T,SCALE,RT)
      IMPLICIT NONE
      EXTERNAL ERROR, FFT, R8TR, ORD1, ORD2, RAN2
c
c     The closer RT is to 1, the more Ns,Nmin & Nmax should be reduced
c     Nmin,Nmax and particularly Nr, are problem dependent
c     NAE (Ne) - the number of previous errors just before the last four
c                temperature changes which were above the error limit
c                simply ends up defining the end temperature.
c
      INTEGER NOP,I,J,K,S,RESTART,
     *        NS,NR,NAE,NK,NMIN,NMAX,NA(50)      ! Marchesi variables
      DOUBLE PRECISION FAC(50),ERR,OPTFAC(50),OPTERR,OLDERR,ERROR,RAN2,
     *        FSUM,F2SUM,ERRLIM,T,RT,TMIN,SCALE(20),A,A1,A2
c
      ERRLIM=1.0E-9                                ! accuracy : O[6-9]
      NR=5                                         !          NR  > 1
      NAE=4                                        ! modified 2x below too
      NS=20                    !              15 !    20 >  NS  > 10
      NMIN=20                  !              15 !    20 > NMIN > 10
      NMAX=300                 !             212 !   300 > NMAX > 150
      NK NMIN                  ! Nk[0] must be NMIN !  NMAX >  NK  > NMIN
      TMIN=0                   ! Tmin(new prob)=0   ! 2.E-4 > TMIN > 1.E-5
      A1=0.5                   ! 1.0
      A2=6.0                   ! 6.0
      S=-6                     ! temporarily use S to initialise RAN2
      ERR=RAN2(S)
      ERR=ERROR(FAC)           ! create an initial error value
c
      OPTERR=ERR               ! define the optimum error for restarts
      DO 1 I=1,NOP
         OPTFAC(I)=FAC(I)      ! define the optimum place for restarts
    1 CONTINUE
c
c     end of set-up ... annealing follows
c
      RESTART=0
    2 CONTINUE
c
c     loop with the current optimisation path, Nr times
c
      DO 6 K=1,NR
      OLDERR=ERR
c
c     reset sum & sum of squares of errors for a given temperature
c
      FSUM=0.
      F2SUM=0.
      S=0
c
c     loop with constant temperature, Nk times
```

```
      DO 5 J=1,NK
c
c     loop with constant step sizes, Ns times
c
      DO 3 I=1,NOP ! null the number of accepted moves in each axis
         NA(I)=0
   3  CONTINUE
c
c debug follows
c     WRITE(17,'(''fac  '',5F9.5|'') (FAC(I),I=1,NOP)
c     WRITE(17,'(''scale'',5F9.5)'') (SCALE(I),I=1,NOP)
c debug ends : plausible, 29/10/92
c
      CALL METROPOLIS
     *     (NOP,NS,NA,FAC,T,SCALE,ERR,OPTERR,OPTFAC,S,FSUM,F2SUM)
c
c debug follows
c     WRITE(17,'(''na..ns'',6I7)'') (NA(I),I=1,NOP),NS
c debug ends : plausible, 29/10/92
c
c Scale is modified as per function 3 (Corana et al) and plotted out by
c scaletest.f . Dependent on NA/NS, scale increases/decreases by up to 3x.
c The scale has been observed to degenerate to zero in NP hard problems,
c suggesting the need to remember the scale of the problem at the last
c sucessful series of iterations...31/10/92.
c
      DO 4 I=1,NOP
         IF (NA(I).GT.(0.6D0*DBLE(NS))) THEN
            SCALE(I)=SCALE(I)*(5.D0*DBLE(NA(I))/DBLE(NS)-2.D0)
         ELSE
            IF (NA(I).LT.(0.4D0*DBLE(NS))) THEN
               SCALE(I)=SCALE(I)*(DBLE(NS)/DBLE(3*NS-5*NA(I)))
            ENDIF
         ENDIF
c
c does the next line correctly interpret 'domain' (Benvenuto) ?
c commented out successfully for simple tests, 31/10/92, REH.
c
c        IF (SCALE(I).GT.DABS(FAC(I)/4.)) SCALE(I)=DABS(FAC(I) 4.)
   4  CONTINUE
c
   5  CONTINUE
c
      IF (DABS(ERR-OLDERR).LT.ERRLIM) THEN
         NAE=NAE-1
      ELSE
         NAE=4                  ! when NE gets to zero the last 4 errors
      ENDIF                     ! must have deviated by less then ERRLIM
c
      IF ((DABS(ERR-OPTERR).LT.ERRLIM).AND.(NAE.LT.1)) RETURN
c
c     calculate A as defined by Benvenuto
c
      A=A2                      ! if not updated because S=0, Nk:=NMAX
      IF (S.GT.0) THEN            ! no allowed accepted perturbations ?
         F2SUM=F2SUM DBLE(S)       ! sum -> average (of squared errors)
         FSUM=FSUM DBLE(S)           ! sum -> average (of errors)
         IF ((F2SUM-FSUM*FSUM).GT.0.) THEN
            A=DSQRT(F2SUM-FSUM*FSUM)/T
         ELSE
            A=A1                     ! if F2=F^2, => ?.. A:=A1 ?(REH)
         ENDIF
      ENDIF
c
c     define Nk using White's criterion
c
      IF (A.LE.A1) THEN
         NK=NMIN
      ELSE
         IF (A.GE.A2) THEN
            NK=NMAX
         ELSE
            NK=NMIN+(NMAX-NMIN)*(A-A1)/(A2-A1)
         ENDIF
      ENDIF
c
      T=RT*T
c
c     record status
c
      WRITE(17,'(3F10.6)') LOG10(  T   + ERRLIM ),
     *                     LOG10(  A   + ERRLIM ),
     *                     LOG10( FSUM + ERRLIM )
c
c debug follows
c
c The following debugging information is written to the standard output
c and may prove useful in understanding the progress/failure of the SA.
c Essential information for the confirmation of global optimisation is
c also written out on unit 17 (file 'status' as defined in 'main').
c
      WRITE(6,'(''K,NAE  '',2I10)')       K,NAE
      WRITE(6,'(''NK,S,T '',2I10,F10.6)') NK,S,T
      WRITE(6,'(''NA     '',5I10)')       (NA(I),I=1,NOP)
      WRITE(6,'(''SCALE  '',5E10.3)')     (SCALE(I),I=1,NOP)
      WRITE(6,'(''FAC    '',5F10.6)')     (FAC(I),I=1,NOP)
      WRITE(6,'(''OPTFAC '',5F10.6)')     (OPTFAC(I),I=1,NOP)
      WRITE(6,'(''ER/RPL '',3E10.2)')      ERR,OPTERR,OLDERR
      WRITE(6,'(''F,F2,A '',3E10.2)')      FSUM,F2SUM,A
      WRITE(6,'(''Log TAF'',3F10.6,/)')
     *     LOG10(T+ERRLIM),LOG10(A+ERRLIM),LOG10(FSUM+ERRLIM)
c
c debug ends
c
c
c     check it's not absolute zero !
c
      IF (T.LT.TMIN) RETURN
c
   6  CONTINUE
c
c     restart with optimal value
c
```

```
      RESTART=RESTART+1
      NAE=4
c
c debug follows
c
      WRITE(6,'(''*****retry***** '',I10,/,''OPTFAC '',5F10.6,/)')
     *                  RESTART,    (OPTFAC(I),I=1,NOP)
c debug ends
c
      DO 7 I=1,NOP
         FAC(I)=OPTFAC(I)
    7 CONTINUE
      ERR OPTERR
      OLDERR=OPTERR
c
      GO TO 2
      END

      SUBROUTINE METROPOLIS
     *          (NOP,NS,NA,FAC,T,SCALE,ERR,OPTERR,OPTFAC,S,FSUM,F2SUM)
      IMPLICIT NONE
      EXTERNAL ERROR, FFT, R8TR, ORD1, ORD2, RAN2
      INTEGER NOP,NS,I,J,K,IDUM,NA(50),S
      DOUBLE PRECISION FAC(50),ERR,NEWFAC(50),NEWERR,OPTFAC(50),OPTERR,
     *                 T, SCALE(20), DELTE,
     *                 ERROR, RAN2,              ! functions used
     *                 FSUM, F2SUM               ! Marchesi variables
      IDUM=1    ! IDUM must be used instead of say 1 or a parameter, as
c               it is modified each time the function is called.
      DO 3 I=1,NS
c
c     perturb
c
         DO 2 J=1,NOP
c        each factor is randomly modified by up to +/- step size:'scale'
            NEWFAC(J)=FAC(J)+2.*SCALE(J)*(0.5-RAN2(IDUM))

            NEWERR=ERROR(NEWFAC)                          ! eval. error
c
C test for improved guess
c
         IF (NEWERR.LT.ERR) THEN                          ! ** ACCEPT **
            NA(J)=NA(J)+1                                 ! increment axis
            S S+1                                         ! increment total
            ERR NEWERR
            FSUM=FSUM+NEWERR                              ! sum the f(x)
            F2SUM=F2SUM+NEWERR*NEWERR                     ! sum the f^2(x)
            FAC(J)=NEWFAC(J)                              ! new coeff
c
c update opterr/optfac if necessary
c
            IF (ERR.LT.OPTERR) THEN
               OPTERR=ERR
               DO 1 K=1,NOP
                  OPTFAC(K)=FAC(K)
    1          CONTINUE
            ENDIF
         ELSE
c
c test for possible usefulness (Metropolis criterion)
c
            DELTE=NEWERR-ERR
c
c IDUM should already be set up by the use of RAN2 above. It should be
c a positive number, ie re-initialisation should not occur, but this
c may be machine dependent. Test above (not here) to verify. OK 4 Suns.
c
            IF (EXP(-DELTE T).GT.RAN2(IDUM)) THEN         ! ** ALLOW **
               NA(J)=NA(J)+1                              ! increment axis
               S=S+1                                      ! increment total
               ERR=NEWERR
               FSUM=FSUM+NEWERR                           ! sum the f(x)
               F2SUM=F2SUM+NEWERR*NEWERR                  ! sum the f^2(x)
               FAC(J)=NEWFAC(J)                           ! new coeff
            ENDIF
         ENDIF                ! ie: otherwise perturbation not accepted
    2 CONTINUE
    3 CONTINUE
c
      RETURN
      END
```

A.3.5 : Pattern Search Noise shaper filter optimisation

Optimisation by pattern search can be used to refine the filters found by the simplex routine without incurring the slow operation of the simulated annealer. 'HJFNSOPT.FOR' is listed below and is well suited to intensively searching a locality near the solution found by the simplex routine. Like the simulated annealer, it uses one stopband suppression figure (ie: flat stopband) but otherwise the same parameter and start files as the simplex routine itself.

HJFNSOPT.FOR

```
      PROGRAM NOISE SHAPING FILTER OPTIMISER ! USING PATTERN SEARCH
      IMPLICIT NONE                          ! NOT MULTI BAND ! FLOATS
      EXTERNAL ERROR, FFT, R8TR, ORD1, ORD2, RAN2
      COMMON /EP  NOP, NOS, C, N, DELTA, WEIGHT
      COMMON  FT/ PII, P7, P7TWO, C22, S22, PI2
      INTEGER NOP,NOS,N,I,J,REPEAT
      DOUBLE PRECISION PII, P7, P7TWO, C22, S22, PI2,
     *                 FAC(50), A(20), C(20), DELTA, WEIGHT,FS, PG,
     *                 STEP, SCALE, RS, ERRLIM, FAIL, ERROR, RAN2
```

```
c     constants for the RADIX 8 FFT...explicitly double precision
      PII = 4.0D+0*ATAN(1.0D+0)
      P7 = 1.0D+0/SQRT(2.0D+0)
      P7TWO = 2.0D+0*P7
      C22 = COS(PII/8.0D+0)
      S22 = SIN(PII/8.0D+0)
      PI2 = 2.0D+0*PII
c
c     variable list : NOP = the number of parameters to be optimised
c                     NOS = the number of specifications to approximate
c                     FAC = the current modification factor set
c                       J = the output file type [0: nothing, 1: Fs, 2: PG]
c     (in main & error) A = the current coefficient set
c                       C = the original coefficient set ( C(1)=1.0! )
c                      FS = the sampling frequency (the input file header)
c                       N = the length of the FFTs used in analysis
c                    DATA = the current N discrete frequency responses' set
c                   DELTA = the passband ripple size (linear)
c                  WEIGHT = the emphasis on satisfying delta over power gain
c                    STEP = the initial move size
c                      NA = the set of numbers of accepted moves at 'scale'
c                      RS = the temperature reduction after NK*NS annealings
c
c nb. FAC(49) and FAC(50) are modified by the error calculating function
c     to the values of the power gain and the weighted mean squared error.
c
c     initialise the random number generator, temporary use of STEP,NOP
      NOP=-6
      STEP=RAN2(NOP)
c
c     define : the first output coefficient to be 1. This is NOT updated !
      A(1)= 1.
c
c     read in the general optimisation parameters a-la-MBNSOPT...
c     typical values : NOP=5,NOS=30,N=512,J=2,DELTA=5.62E-3,WEIGHT=1.E+6
c     & set up the pattern search parameters (see also subroutine HKJV)
c     typical values :    STEP : ~ max coeff/2,
c                       ERRLIM :   O[ -6 ~ -9 ],
c                         1.0 > RT > 0.99
c
      OPEN(UNIT=15,FILE='hjpar',STATUS='OLD')
c
      READ(15,1001) NOP
      READ(15,1001) NOS
      READ(15,1001) N
      READ(15,1001) J
      READ(15,1003) DELTA
      READ(15,1003) WEIGHT
      READ(15,1003) STEP
      READ(15,1003) SCALE
      READ(15,1001) REPEAT
      READ(15,1003) ERRLIM
      READ(15,1003) RS
c
      CLOSE(UNIT=15)
c
      OPEN(UNIT=15,FILE='start',STATUS='OLD')
c
c     read in the sample rate & the starting coefficients as directed by 'J'
      IF (J.EQ.1) THEN
          READ(15,1002)FS
      ENDIF
      IF (J.EQ.2) THEN
          READ(15,1002)PG
      ENDIF
      DO 1 I=1,NOP+1
         READ(15,1003) C(I)
    1 CONTINUE
c
      CLOSE(UNIT=15)
c
c     start with the coefficient modification factors at unity
c     and the scale of parameter moves as read into scale(1).
      DO 2 I=1,NOP
         FAC(I)=1.0
    2 CONTINUE
c
c     define a record file
c
      OPEN(UNIT=17,FILE='status',STATUS='UNKNOWN')  ! use debug unit
      CALL HKJV(NOP,FAC,STEP,RS,ERRLIM,SCALE,REPEAT)
      CLOSE(UNIT=17)                                ! if opened above
c
c     calculate the new estimates of the coefficients
      DO 3 I=1,NOP
         A(I+1)=FAC(I)*C(I+1)
    3 CONTINUE
      FAIL=ERROR(FAC)                    ! dummy call to evaluate the error
c
c     open the output file on unit 16
      OPEN(UNIT=16,FILE='bestyet',STATUS='UNKNOWN')
c
c     output the sampling frequency and new coefficients as directed by 'J'
      IF (J.EQ.1) THEN
          WRITE(16,1002)FS
      ENDIF
      IF (J.EQ.2) THEN
          WRITE(16,1002)FAC(49)
      ENDIF
      DO 4 I=1,NOP+1
         WRITE(16,1003)A(I)
    4 CONTINUE
      IF (J.EQ.0) THEN
          WRITE(16,*)'Stop - Program terminated.'
      ENDIF
      CLOSE(UNIT=16)
c
      WRITE(6,*)'passband failure measured at ',FAC(50)
 1001 FORMAT(I5)
 1002 FORMAT(E15.4)
 1003 FORMAT(E15.9)
      STOP
      END
```

```
      SUBROUTINE HKJV(NOP,FAC,OLDSTEP,RS,ERRLIM,SCALE,REPEAT)
      IMPLICIT NONE
      INTEGER I,J,NOP,IDUM,REPEAT
      DOUBLE PRECISION SCALE,STEP,OLDSTEP,RS,ERRLIM,ERR,OPTERR,BESERR,
     *                 FAC(50),OPTFAC(50),OLDFAC(50),BESFAC(50),ERROR,RAN2
c
c     Randomised pattern search; ref: Hooke & Jeeves (J.ACM,1961)
c     Variables:
c             s : functional value before this move (OPTERR)
c                (smallest value so far from the set of exploratory moves)
c             k : counter (I,J,K)
c             K : number of coordinates for the points (NOP)
c        coord : array of the current coordinates (FAC)
c        delta : current step size (STEP)
c         temp : temporary functional value
c     lastbase : coordinates for the previous base point (OLDFAC)
c     currbase : coordinates for the current base point (OPTFAC)
c     nextbase : coordinates resulting for the current move (FAC)
c       redfac : reduction factor for the step size (RS)
c       errlim : smallest step size (ERRLIM)
c
      DO 1 I=1,NOP
         OPTFAC(I)=FAC(I)
         BESFAC(I)=FAC(I)
         OLDFAC(I)=FAC(I)
    1 CONTINUE
      OPTERR=ERROR(OPTFAC)
      BESERR=OPTERR
      STEP OLDSTEP              ! memorise the starting size of step
      IDUM=6                   ! define the random number generator arg.
c
c debug follows
c
c     WRITE(17,'(''FAC'',5F10.6)') (FAC(I),I=1,NOP)
c     WRITE(17,'(''OPT'',5F10.6)') (OPTFAC(I),I=1,NOP)
c     WRITE(17,'(''OLD'',5P10.6)') (OLDFAC(I),I=1,NOP)
c     WRITE(17,'(''OPTERR'',F10.5,'' STEP'',F10.5,'' RS'',F10.5)')
c    *             OPTERR,STEP,RS
c debug ends
c
c repeatedly pattern search from +/-scale/2*random from best yet found
c
      DO 11 J=1,REPEAT
      OPEN(UNIT=18,FILE='stop',STATUS='OLD',ERR=2)
      RETURN
c
c ----------------beginning of one pattern search--------------
    2 CONTINUE
c
      ERR=OPTERR
      CALL EXPLORE(NOP,FAC,STEP,ERR)
      IF (ERR.LT.OPTERR) THEN
          DO 3 I=1,NOP
             OLDFAC(I)=OPTFAC(I)
             OPTFAC(I)=FAC(I)
             OPTERR=ERR
    3     CONTINUE
      ELSE
          IF (STEP.LT.ERRLIM) THEN ! assuming a better set of Fac's found
              DO 4 I=1,NOP
                 FAC(I)=OPTFAC(I)
    4         CONTINUE
              ERR=OPTERR
              GOTO 8
          ELSE
              STEP=RS*STEP
              GOTO 2
          ENDIF
      ENDIF
    5 CONTINUE
      CALL PATTERN(NOP,OPTFAC,OLDFAC,FAC)
      CALL EXPLORE NOP,FAC,STEP,ERR)
      IF (ERR.LT.OPTERR) THEN
          DO 6 I=1,NOP
             OLDFAC(I)=OPTFAC(I)
             OPTFAC(I)=FAC(I)
             OPTERR=ERR
    6     CONTINUE
          GOTO 5
      ELSE
          DO 7 I=1,NOP
             FAC(I)=OPTFAC(I)
    7     CONTINUE
          GOTO 2
      ENDIF
c ---------------end of one pattern search------------
c
c store the best finish point before restarting
c
    8 IF (ERR.LT.BESERR) THEN
         DO 9 I=1,NOP
            BESFAC(I)=FAC(I)
    9    CONTINUE
         BESERR=ERR
      ENDIF
c
c report to status file
c
      WRITE(17,'(I5,E13.6,50F10.6)') J,ERR,(FAC(I),I=1,NOP)
c
c debug follows, 'before'
c
      WRITE(6,'(''finish '',I4,50F10.6)') J,(FAC(I),I=1,NOP)
c
c debug ends
c
      STEP=OLDSTEP              ! reset step size
      DO 10 I=1,NOP             ! re-initialise variables
         FAC(I)=BESFAC(I)+SCALE*(RAN2(IDUM)-0.5D0)
         OLDFAC(I)=FAC(I)
         OPTFAC(I)=FAC(I)
         OPTERR=ERROR(OPTFAC)
   10 CONTINUE                  ! don't re-initialise besterr/besfac()
```

```
c debug follows, 'after'
c
      WRITE(6,'(''restart'',I4,50F10.6)') J+1,(FAC(I),I=1,NOP)
c
c debug ends
c
  11  CONTINUE                ! end of restart loop
c
      DO 12 I=1,NOP
         FAC(I)=BESFAC(I)     ! return the best of 100 restarts
  12  CONTINUE
      RETURN
      END

      SUBROUTINE EXPLORE(NOP,FAC,STEP,OPTERR)
c
c Call explore with a copy of opterr in err, passed as the last argument
c this way opterr outside isn't changed and can be used for comparison.
c
      IMPLICIT NONE
      INTEGER I,J(50),K,NOP
      DOUBLE PRECISION FAC(50),ERR,STEP,OPTERR,TRYFAC(50),ERROR
      CALL SHUFL(J,NOP)           ! random order of 1st modification
      DO 1 K=1,NOP
         I=J(K)
         TRYFAC(I)=FAC(I)
         FAC(I)=FAC(I)+STEP       ! try +ve
         ERR=ERROR(FAC)
         IF (ERR.LE.OPTERR) THEN
             OPTERR=ERR
             TRYFAC(I)=FAC(I)
         ENDIF
         FAC(I)=FAC(I)-2.D0*STEP ! check -ve
         ERR=ERROR(FAC)
         IF (ERR.LE.OPTERR) THEN
             OPTERR=ERR
             TRYFAC(I)=FAC(I)
         ENDIF
         FAC(I)=TRYFAC(I)         ! accept the best of -ve,+ve,original
   1  CONTINUE
      RETURN
      END

      SUBROUTINE PATTERN(NOP,OPTFAC,OLDFAC,FAC)
      IMPLICIT NONE
      INTEGER I,NOP
      DOUBLE PRECISION OPTFAC(50),OLDFAC(50),FAC(50)
      DO 1 I=1,NOP
         FAC(I)=2.D0*OPTFAC(I)-OLDFAC(I)
   1  CONTINUE
      RETURN
      END

      SUBROUTINE SHUFL(J,NOP)
      IMPLICIT NONE
      INTEGER I,J(50),K,NOP,IDUM,SWAP
      DOUBLE PRECISION RAN2
      IDUM=6
      DO 1 I=1,NOP
         J(I)=I
   1  CONTINUE
      DO 2 I=NOP,1,-1
         SWAP=INT(RAN2(IDUM)*DREAL(I))+1
         K=J(I)
         J(I)=J(SWAP)
         J(SWAP)=K
   2  CONTINUE
      RETURN
      END
```

A.3.6 : FIR filter coefficient truncation

Programme 'NSFROUND.FOR' is listed overleaf, and allows the truncation of existing FIR filters by noise shaping the impulse response. In general, passband ripple can be far larger than stopband ripple, so noise shaping errors introduced in quantising coefficients to lie outside the stopband is useful. Tests in chapter 4 demonstrated that significant benefits can be gained over simple truncation or rounding of coefficients. In this programme, noise shaper states are initialised with noise before processing the NTF impulse response itself. By repeatedly re-initialising these states until the first coefficient is unity, re-scaling of the output (which destroys the truncation just achieved) can be avoided.

NSFROUND.FOR

```
      PROGRAM TO_TRUNCATE_COEFFICIENTS ! OF AN FIR FILTER
      EXTERNAL RAN2
C
      DOUBLE PRECISION A(1000), C(100), ERROR(100)
      INTEGER NUMBIT, ORDER, OVERRUN, OFFSET, B(1000)
C
C -----read & check the header-----
      READ(5,*) NUMBIT, ORDER, OVERRUN
      IF (NUMBIT.GT.20) THEN
         WRITE(0,*) ' 20 bit quantisation max. '
         STOP
         ENDIF
```

```
      IF (ORDER.GT.100) THEN
         WRITE(0,*) ' 100 tap filters max. '
         STOP
         ELSE
            OFFSET=5*ORDER
         ENDIF
      IF (OVERRUN.GT.4*ORDER) THEN
         WRITE(0,*) ' filter growth 300% max. '
         STOP
         ENDIF
       J=1
C -----work out 2^numbit-----
      DO 1 I=1, NUMBIT
         J=J*2
1        CONTINUE
C -----repeat ESS until the first coefficient is unity-----
      DO 6 L=1,100
C
C -----initialise the time domain array-----
      DO 2 I=1,2*OFFSET
         A(I)=RAN2(1-L)/J            ! first call => self initialises
2        CONTINUE
C
C -----read the filter coefficients (ie impulse response)-----
      DO 3 I=1, ORDER+1
         IF (L.EQ.1) READ(5,*) C(I) ! prevent re-read
         A(I+OFFSET)=J*C(I)
         ERROR(I)=RAN2(0)            ! initialise the error
3        CONTINUE
C
C -----check the filter makes sense-----
      IF (C(1).NE.-1.0) THEN
         WRITE(0,*) L, C(1)
         WRITE(0,*) ' -1 expected as the first coefficient.'
         STOP
         ENDIF
C
C -----operate the noise shaper------
      DO 5 I=1,2*OFFSET
         FEEDBACK 0.0
         DITHER RAN2(0)-0.5
         DO 4 K=ORDER, 1, -1                    ! accumulate ESS signal
            FEEDBACK=FEEDBACK+ERROR(K)*C(K+1)
            ERROR(K+1)=ERROR(K)
4           CONTINUE
         A(I)=A(I)+FEEDBACK                     ! evaluate signals
         B(I)=IDNINT(A(I)+DITHER)               ! rounding
         ERR R(1)=A(I)-B(I)
5        CONTINUE
C
C --------check for endless searching-----
         IF (L.EQ.100) THEN
            L=2
            WRITE(0,*) ' the first filter coefficient isn''t one yet'
            PAUSE
            ENDIF
         IF (-1*B(OFFSET+1).EQ.J) L=100          ! terminate loop
6        CONTINUE
C
C --------write out the header & new filter-----
         WRITE(6,*) NUMBIT, ORDER, OVERRUN
         DO 7 I=1,ORDER+OVERRUN+1
            WRITE(6,*) B(I+OFFSET)
7           CONTINUE
         STOP
      END
```

A.3.7 : Analogue filter nodal analysis

In the design of analogue filters for recovery and carrier suppression, classical designs need to be re-assessed to include parasitics introduced by non-ideal components and nearest value components. Programme 'RESPONSE.FOR' was adapted for this purpose, using a nodal representation of the implementation of the filter and producing the magnitude and phase responses.

RESPONSE.FOR

```
      PROGRAM COMPLEX AMPLITUDE AND PHASE ANALYSER
C
C     APADTED BY ROD.HIORNS, 22 4 90 & 13 10 93
C     FROM A PROGRAM BY JOHN WATKINS: JWCAPU.FOR 02/2/89
C
      COMMON YY(35,35),KA(50),KB(50),KC(50),KD(50),KE(50),PA(50)
     1,PB(50),NTOP,M,FREQ,FR(50),INDIC,QQQ
      DIMENSION GC(50),AG(50),BG(50),GCM(50),AGM(50),BGM(50)
      CHARACTER*60 SAMPL
      CHARACTER*1 ANS
      COMPLEX YY,A,DEL,ZIN,ZOT
      OPEN(UNIT=16,FILE='INPUT.DAT',STATUS='UNKNOWN')
      OPEN(UNIT=17,FILE='OUTPUT.DAT',STATUS='UNKNOWN')
      READ(16,20)SAMPL
20    FORMAT(A)
      WRITE(6,21)SAMPL
21    FORMAT(1H1,A)
      READ(16,*)NTOP,M,NGAIN,BEX,DEX,ITAG
      WRITE(6,2)NTOP,M,NGAIN,BEX,DEX
    2 FORMAT(1H , 5HNTOP=,I2 1H ,    16HNO. OF BRANCHES=,I2/1H ,
     + 17HGAIN OFFSET(DB.)=,I3 1H ,   4HBEX=,F7.4/1H ,    4HDEX=,F9.3//)
      QQQ=1000.
      DO   5 J=1,M
      READ(16,*) KE(J),KA(J),KB(J),KC(J),KD(J),PA(J),PB(J)
      WRITE(6,4) KE(J),KA(J),KB(J),KC(J),KD(J),PA(J),PB(J)
    4 FORMAT(1H ,I1,1X,4(I3),2(E12.4))
```

```
 5 CONTINUE
41 FORMAT(//
  +1H ,5X,9HFREQUENCY,6X,21HINPUT IMPEDANCE(OHMS),9X,22HOUTPUT IMPEDA
  +NCE(OHMS)/)
   EX=BEX
   DO 6 J=1,50
   GC(J)=EX
   FR(J)=EX
   EX=EX+DEX
 6 CONTINUE
   DO 9 KK=1,50
   FREQ=FR(KK)
   CALL MATRIX
   CALL TWOTWO
   A=-YY(2,1)/YY(2,2)
   AA=CABS(A)
   A1=20.*ALOG10(AA)
   AG(KK)=A1
   AR=REAL(A)
   AI=AIMAG(A)
   PH=(ATAN2(AI,AR))*57.2958
   BG(KK)=PH
   DEL=YY(1,1)*YY(2,2)-YY(1,2)*YY(2,1)
   ZIN=YY(2,2)/DEL
   ZOT=YY(1,1)/DEL
   RIN=REAL(ZIN)
   ROT=REAL(ZOT)
   CIN=AIMAG(ZIN)
   COT=AIMAG(ZOT)
   IF((KK/ITAG-(KK-1)/ITAG) .EQ. 0) GO TO 9
 9 CONTINUE
   DO 92 KK=1,50
   WRITE(17,91)KK,FR(KK),AG(KK),BG(KK)
91 FORMAT(1H ,I2,2X,F10.0,6X,F10.2,7X,F10.2)
92 CONTINUE
   STOP
   END

   SUBROUTINE DISTRC
   COMMON YY(35,35),KA(50),KB(50),KC(50),KD(50),KE(50),PA(50)
  1,PB(50),NTOP,M,FREQ,FR(50),INDIC,QQQ
   COMPLEX YY,ARG2,DEZ,ARG,ZED,YD,YAF,CCOSH,CSINH
   I=INDIC
   NT=NTOP+1
   NA=KA(I)
   NB=KB(I)
   NC=KC(I)
   ND=KD(I)
   W=2.*3.1415927*FREQ
   U1=W*PA(I)*PB(I)
   V1=-PA(I)/(W*PB(I))
   ARG2=CMPLX(0.,U1)
   DEZ=CMPLX(0.,V1)
   ARG=CSQRT(ARG2)
   ZED=CSQRT(DEZ)
   YD=(CCOSH(ARG)-1.)/(ZED*CSINH(ARG))
   YAF=1./(ZED*CSINH(ARG))
   YY(NA,NA)=YY(NA,NA)+YAF
   YY(NA,NB)=YY(NA,NB)-YAF
   YY(NB,NA)=YY(NB,NA)-YAF
   YY(NB,NB)=YY(NB,NB)+YAF
   YY(NA,NA)=YY(NA,NA)+YD
   YY(NA,NC)=YY(NA,NC)-YD
   YY(NC,NA)=YY(NC,NA)-YD
   YY(NC,NC)=YY(NC,NC)+YD
   YY(NB,NB)=YY(NB,NB)+YD
   YY(NB,NC)=YY(NB,NC)-YD
   YY(NC,NB)=YY(NC,NB)-YD
   YY(NC,NC)=YY(NC,NC)+YD
   RETURN
   END

   SUBROUTINE TLIN
   COMMON YY(35,35),KA(50),KB(50),KC(50),KD(50),KE(50),PA(50)
  1,PB(50),NTOP,M,FREQ,FR(50),INDIC,QQQ
   COMPLEX YY,TERME,TERMF,TERMG,YD,YAF
   I=INDIC
   NT=NTOP+1
   NA=KA(I)
   NB=KB(I)
   NC=KC(I)
   ND=KD(I)
   Z0=PA(I)
   EL=PB(I)
   BETA=2.*3.1415927*FREQ/2.997925E8
   ALFA=BETA/(2.*QQQ)
   PHA=BETA*EL
   PHB=ALFA*EL
   TERMA=COSH(PHB)*COS(PHA)
   TERMB=SINH(PHB)*SIN(PHA)
   TERMC=SINH(PHB)*COS(PHA)
   TERMD=COSH(PHB)*SIN(PHA)
   TERME=CMPLX(-1.,0.)
   TERMF=CMPLX(TERMA,TERMB)
   TERMG=CMPLX(TERMC,TERMD)
   YD=(TERMF/TERMG)+(TERME/TERMG)
   YAF=-TERME/TERMG
   YD=YD/Z0
   YAF=YAF/Z0
   YY(NA,NA)=YY(NA,NA)+YAF
   YY(NA,NB)=YY(NA,NB)-YAF
   YY(NB,NA)=YY(NB,NA)-YAF
   YY(NB,NB)=YY(NB,NB)+YAF
   YY(NA,NA)=YY(NA,NA)+YD
   YY(NA,NC)=YY(NA,NC)-YD
   YY(NC,NA)=YY(NC,NA)-YD
   YY(NC,NC)=YY(NC,NC)+YD
   YY(NB,NB)=YY(NB,NB)+YD
   YY(NB,NC)=YY(NB,NC)-YD
   YY(NC,NB)=YY(NC,NB)-YD
   YY(NC,NC)=YY(NC,NC)+YD
   RETURN
   END
```

```
      SUBROUTINE MATRIX
      COMMON YY(35,35),KA(50),KB(50),KC(50),KD(50),KE(50),PA(50)
     1,PB(50),NTOP,M,FREQ,FR(50),INDIC,QQQ
      COMPLEX YY,YE
      DATA KFDNR,KDL,KTL,KGC,KRL/5,4,3,2,1/
      OMEGA=6.28319*FREQ
      NT=1+NTOP
      DO 2 J=1,NT
      DO 1 I=1,NT
      YY(I,J)=(0.,0.)
    1 CONTINUE
      ZT=.1
    2 CONTINUE
      DO 4 INDIC=1,M
      I=INDIC
      NA KA(I)
      NB=KB(I)
      NC=KC(I)
      ND=KD(I)
      NE=KE(I)
      IF(NE.EQ.KTL) GO TO 30
      IF(NE.EQ.KDL) GO TO 40
      IF(NE.EQ.KFDNR) GO TO 50
      GE PA(I)
      BE=OMEGA*PB(I)
      YE CMPLX(GE,BE)
      IF(NE.EQ.KGC)GO TO 3
      T=GE*GE+BE*BE
      GE=GE/T
      BE=-BE/T
      YE=CMPLX(GE,BE)
      GO TO 3
   30 CALL TLIN
      GO TO 4
   40 CALL DISTRC
      GO TO 4
   50 GE=-(OMEGA**2)*PA(I)
      BE=0.0
      YE=CMPLX(GE,BE)
    3 YY(NC,NA)=YY(NC,NA)+YE
      YY(NC,NB)=YY(NC,NB)-YE
      YY(ND,NA)=YY(ND,NA)-YE
      YY(ND,NB)=YY(ND,NB)+YE
    4 CONTINUE
      RETURN
      END

      SUBROUTINE TWOTWO
      COMMON YY(35,35),KA(50),KB(50),KC(50),KD(50),KE(50),PA(50)
     1,PB(50),NTOP,M,FREQ,FR(50),INDIC,QQQ
      COMPLEX YY,TEMPC,YC,QJ,QI
      N NTOP
    1 IF(N.EQ.2) RETURN
      PIV=0.
      DO 2 I=3,N
      Y=CABS(YY(I,N))
      IF(PIV.GT.Y)GO TO 2
      PIV=Y
      K=I
      YC=YY(I,N)
    2 CONTINUE
      IF(PIV.LT.1.E-36)GO TO 7
      IF(K.EQ.N)GO TO 4
      DO 3 I=1,N
      TEMPC=YY(K,I)
      YY(K,I)=YY(N,I)
      YY(N,I)=TEMPC
    3 CONTINUE
    4 K=N-1
      DO 10 I=1,K
      DO 10 J=1,K
      IF((YY(J,N).EQ.(0.,0.)).OR.(YY(N,I).EQ.(0.,0.)))GO TO 10
      QJ=YY(J,N) YY(N,N)
      QI=YY(N,I)/YY(N,N)
      RQJ=CABS(QJ)
      RQI=CABS(QI)
      DJ=ABS(1.-RQJ)
      DI=ABS(1.-RQI)
      IF (DJ .LT. DI) GO TO 110
      YY(J,I)=YY(J,I)-QI*YY(J,N)
      GO TO 10
  110 YY(J,I)=YY(J,I)-QJ*YY(N,I)
   10 CONTINUE
      N=K
      GO TO 1
    7 WRITE(17,'(1H0,60X, 27HNO SOLUTION. DIAGONAL ZERO.)')
      STOP
      END
```

Appendix 4 : Associated Functions used in A1 - A3.

A.4.1 : Pseudo random number generator

For many of the routines in appendices 1-3, a random number generator with high sample to sample independence and white spectral characteristics is required. Listed below is 'RAN2.FUN' as implemented in the book, 'Numerical Recipes in Fortran', and this is required as an external function for QENS.FOR, SANSOPT.FOR, HJFNSOPT.FOR AND NSFROUND.FOR.

RAN2.FUN

```
      FUNCTION RAN2(IDUM)              ! from the Numerical Recipes Book
C
C returns uniformly distributed random real numbers between 0 and 1
C set IDUM < 0 to re-initialise at any time
C
      PARAMETER (M=714025, IA=1366, IC=150889, RM=1./M)
      DIMENSION IR(97)
      DATA IFF /0/  ! 1st time, initialise the random number generator
      IF (IDUM.LT.0.OR.IFF.EQ.0) THEN
         IFF=1
         IDUM=MOD(IC-IDUM,M)
         DO 1 J=1,97                        ! initialise the shuffle table
            IDUM=MOD(IA*IDUM+IC,M)
            IR(J)=IDUM
1           CONTINUE
         IDUM=MOD(IA*IDUM+IC,M)
         IY=IDUM
         ENDIF
      J=1+(97*IY)/M                 ! entry point without initialisation
      IF ((J.GT.97).OR.(J.LT.1)) PAUSE
      IY=IR(J)
      RAN2=IY*RM
      IDUM=MOD(IA*IDUM+IC,M)
      IR(J)=IDUM
      RETURN
      END
```

A.4.2 : High accuracy, large argument, high order Bessel function

Finding the result of a large argument Bessel function is prone to inaccuracy, especially for high order Bessel functions. For the evaluation of theoretical performance that could be expected from various modulation types (as analysed in appendix 8), such a routine was required with accuracy comparable to the system resolution (better than 1 part per million). For this a special interpretation of the expansion of a series for Bessel functions of the first kind was implemented, as listed below:

BESSEL.UNT

```
UNIT bessel;
{$N+}
INTERFACE
FUNCTION J(n:INTEGER;x:DOUBLE):DOUBLE;

IMPLEMENTATION
FUNCTION J;
VAR modifier, sum, temp_ext : EXTENDED ;
    index                   : INTEGER  ;
BEGIN
   temp_ext := 1.0 ;
   sum      := 1.0 ;
   index    := 2   ;
   IF n=0 THEN { do the zero evaluation }
   BEGIN
      REPEAT
         modifier := SQR(x/index);
         temp_ext := temp_ext * modifier;
         IF (index MOD 4 = 0) THEN sum := sum + temp_ext
                              ELSE sum := sum - temp_ext;
         INC(index,2);
      UNTIL ((temp_ext < 1.0e-20) OR (index > 100));
   END
   ELSE { do the non zero evaluation }
   BEGIN
      REPEAT
         modifier := (SQR(x/2.0) * 2.0/index)/(index/2.0 + n);
         temp_ext := temp_ext * modifier;
         IF (index MOD 4 = 0) THEN sum := sum + temp_ext
                              ELSE sum := sum - temp_ext;
         INC(index,2);
      UNTIL ((temp_ext < 1.0e-10) OR (index > 100));
      FOR index := n DOWNTO 1 DO sum := sum * x/(2.0*index);
   END;
   J := sum ;
END; { Function J }

{ INITIALIZATION }
BEGIN
END.
```

A.4.3 : Radix 8, real data FFT with in-place re-ordering

A fast Fourier transform is required for all of the optimisers described in the last section and the minimum phase conversion routine MPC.FOR. To ensure fast evaluation and retain some flexibility in size without too much programme overhead (storage and instructions), a radix 8 FFT was chosen. This was selected for its speed and limited use of storage space, since it provides in-place re-ordering and handles only real input data.

FFT.SUB

```
      SUBROUTINE FFT(B, NFFT) ! modified from the IEEE programs for DSP
      DOUBLE PRECISION T, B(2)
      M  = NINT(LOG(NFFT)/LOG(2))
      NN = 1
      DO 80 IT=1,3
        NN = NN*8
        INT = NFFT/NN
        CALL R8TR(INT, NN, B(1), B(INT+1), B(2*INT+1), B(3*INT+1),
     *       B(4*INT+1), B(5*INT+1), B(6*INT+1), B(7*INT+1), B(1),
     *       B(INT+1), B(2*INT+1), B(3*INT+1), B(4*INT+1), B(5*INT+1),
     *       B(6*INT+1), B(7*INT+1))
   80 CONTINUE
      CALL ORD1(M, B)
      CALL ORD2(M, B)
      T = B(2)
      B(2) = 0.
      B(NFFT+1) = T
      B(NFFT+2) = 0.
      DO 100 I=4,NFFT,2
        B(I) = -B(I)
  100 CONTINUE
      RETURN
      END

      SUBROUTINE R8TR(INT, NN, BR0, BR1, BR2, BR3, BR4, BR5, BR6, BR7,
     *    BI0, BI1, BI2, BI3, BI4, BI5, BI6, BI7)
      COMMON  PT/ PII, P7, P7TWO, C22, S22, PI2
      DIMENSION L(15)
      EQUIVALENCE (L15,L(1)), (L14,L(2)), (L13,L(3)), (L12,L(4)),
     *    (L11,L(5)), (L10,L(6)), (L9,L(7)), (L8,L(8)), (L7,L(9)),
     *    (L6,L(10)), (L5,L(11)), (L4,L(12)), (L3,L(13)), (L2,L(14)),
     *    (L1,L(15))
      DOUBLE PRECISION PII, P7, P7TWO, C22, S22, PI2,
     *    ARG, PI, PIOVN, PR, TH2,
     *    C1,C2,C3,C4,C5,C6,C7,
     *    S1,S2,S3,S4,S5,S6,S7,
     *    T0,T1,T2,T3,T4,T5,T6,T7,
     *    TR0,TR1,TR2,TR3,TR4,TR5,TR6,TR7,
     *    TI0,TI1,TI2,TI3,TI4,TI5,TI6,TI7,
     *    BR0(2),BR1(2),BR2(2),BR3(2),BR4(2),BR5(2),BR6(2),BR7(2),
     *    BI0(2),BI1(2),BI2(2),BI3(2),BI4(2),BI5(2),BI6(2),BI7(2)
      L(1) = NN 8
      DO 40 K=2,15
        IF (L(K-1)-2) 10, 20, 30
   10   L(K-1) = 2
   20   L(K) = 2
        GO TO 40
   30   L(K) = L(K-1)/2
   40 CONTINUE
      PIOVN = PII/FLOAT(NN)
      JI = 3
      JL = 2
      JR = 2
      DO 120 J1=2,L1,2
      DO 120 J2=J1,L2,L1
      DO 120 J3=J2,L3,L2
      DO 120 J4=J3,L4,L3
      DO 120 J5=J4,L5,L4
      DO 120 J6=J5,L6,L5
      DO 120 J7=J6,L7,L6
      DO 120 J8=J7,L8,L7
      DO 120 J9=J8,L9,L8
      DO 120 J10=J9,L10,L9
      DO 120 J11=J10,L11,L10
      DO 120 J12=J11,L12,L11
      DO 120 J13=J12,L13,L12
      DO 120 J14=J13,L14,L13
      DO 120 JTHET=J14,L15,L14
        TH2 = JTHET - 2
        IF (TH2) 50, 50, 90
   50   DO 60 K=1,INT
          T0 = BR0(K) + BR4(K)
          T1 = BR1(K) + BR5(K)
          T2 = BR2(K) + BR6(K)
          T3 = BR3(K) + BR7(K)
          T4 = BR0(K) - BR4(K)
          T5 = BR1(K) - BR5(K)
          T6 = BR2(K) - BR6(K)
          T7 = BR3(K) - BR7(K)
          BR2(K) = T0 - T2
          BR3(K) = T1 - T3
          T0 = T0 + T2
          T1 = T1 + T3
          BR0(K) = T0 + T1
          BR1(K) = T0 - T1
          PR = P7*(T5-T7)
          PI = P7*(T5+T7)
          BR4(K) = T4 + PR
          BR7(K) = T6 + PI
          BR6(K) = T4 - PR
          BR5(K) = PI - T6
   60   CONTINUE
```

```
      IF (NN-8) 120, 120, 70
70    K0 = INT*8 + 1
      KL = K0 + INT - 1
      DO 80 K=K0,KL
        PR = P7*(BI2(K)-BI6(K))
        PI = P7*(BI2(K)+BI6(K))
        TR0 = BI0(K) + PR
        TI0 = BI4(K) + PI
        TR2 = BI0(K) - PR
        TI2 = BI4(K) - PI
        PR = P7*(BI3(K)-BI7(K))
        PI = P7*(BI3(K)+BI7(K))
        TR1 = BI1(K) + PR
        TI1 = BI5(K) + PI
        TR3 = BI1(K) - PR
        TI3 = BI5(K) - PI
        PR = TR1*C22 - TI1*S22
        PI = TI1*C22 + TR1*S22
        BI0(K) = TR0 + PR
        BI6(K) = TR0 - PR
        BI7(K) = TI0 + PI
        BI1(K) = PI - TI0
        PR = -TR3*S22 - TI3*C22
        PI = TR3*C22 - TI3*S22
        BI2(K) = TR2 + PR
        BI4(K) = TR2 - PR
        BI5(K) = TI2 + PI
        BI3(K) = PI - TI2
80    CONTINUE
      GO TO 120
90    ARG = TH2*PIOVN
      C1 = COS(ARG)
      S1 = SIN(ARG)
      C2 = C1**2 - S1**2
      S2 = C1*S1 + C1*S1
      C3 = C1*C2 - S1*S2
      S3 = C2*S1 + S2*C1
      C4 = C2**2 - S2**2
      S4 = C2*S2 + C2*S2
      C5 = C2*C3 - S2*S3
      S5 = C3*S2 + S3*C2
      C6 = C3**2 - S3**2
      S6 = C3*S3 + C3*S3
      C7 = C3*C4 - S3*S4
      S7 = C4*S3 + S4*C3
      INT8 = INT*8
      J0 = JR*INT8 + 1
      K0 = JI*INT8 + 1
      JLAST = J0 + INT - 1
      DO 100 J=J0,JLAST
        K = K0 + J - J0
        TR1 = BR1(J)*C1 - BI1(K)*S1
        TI1 = BR1(J)*S1 + BI1(K)*C1
        TR2 = BR2(J)*C2 - BI2(K)*S2
        TI2 = BR2(J)*S2 + BI2(K)*C2
        TR3 = BR3(J)*C3 - BI3(K)*S3
        TI3 = BR3(J)*S3 + BI3(K)*C3
        TR4 = BR4(J)*C4 - BI4(K)*S4
        TI4 = BR4(J)*S4 + BI4(K)*C4
        TR5 = BR5(J)*C5 - BI5(K)*S5
        TI5 = BR5(J)*S5 + BI5(K)*C5
        TR6 = BR6(J)*C6 - BI6(K)*S6
        TI6 = BR6(J)*S6 + BI6(K)*C6
        TR7 = BR7(J)*C7 - BI7(K)*S7
        TI7 = BR7(J)*S7 + BI7(K)*C7
C
        T0 = BR0(J) + TR4
        T1 = BI0(K) + TI4
        TR4 = BR0(J) - TR4
        TI4 = BI0(K) - TI4
        T2 = TR1 + TR5
        T3 = TI1 + TI5
        TR5 = TR1 - TR5
        TI5 = TI1 - TI5
        T4 = TR2 + TR6
        T5 = TI2 + TI6
        TR6 = TR2 - TR6
        TI6 = TI2 - TI6
        T6 = TR3 + TR7
        T7 = TI3 + TI7
        TR7 = TR3 - TR7
        TI7 = TI3 - TI7
C
        TR0 = T0 + T4
        TI0 = T1 + T5
        TR2 = T0 - T4
        TI2 = T1 - T5
        TR1 = T2 + T6
        TI1 = T3 + T7
        TR3 = T2 - T6
        TI3 = T3 - T7
        T0 = TR4 - TI6
        T1 = TI4 + TR6
        T4 = TR4 + TI6
        T5 = TI4 - TR6
        T2 = TR5 - TI7
        T3 = TI5 + TR7
        T6 = TR5 + TI7
        T7 = TI5 - TR7
        BR0(J) = TR0 + TR1
        BI7(K) = TI0 + TI1
        BI6(K) = TR0 - TR1
        BR1(J) = TI1 - TI0
        BR2(J) = TR2 - TI3
        BI5(K) = TI2 + TR3
        BI4(K) = TR2 + TI3
        BR3(J) = TR3 - TI2
        PR = P7*(T2-T3)
        PI = P7*(T2+T3)
        BR4(J) = T0 + PR
        BI3(K) = T1 + PI
        BI2(K) = T0 - PR
        BR5(J) = PI - T1
```

```
          PR = -P7*(T6+T7)
          PI = P7*(T6-T7)
          BR6(J) = T4 + PR
          BI1(K) = T5 + PI
          BI0(K) = T4 - PR
          BR7(J) = PI - T5
  100     CONTINUE
          JR = JR + 2
          JI = JI - 2
          IF (JI-JL) 110, 110, 120
  110     JI = 2*JR - 1
          JL = JR
  120   CONTINUE
        RETURN
        END

        SUBROUTINE ORD1(M, B)
        DOUBLE PRECISION T, B(2)
        K = 4
        KL = 2
        N = 2**M
        DO 40 J=4,N,2
          IF (K-J) 20, 20, 10
   10     T = B(J)
          B(J) = B(K)
          B(K) = T
   20     K = K - 2
          IF (K-KL) 30, 30, 40
   30     K = 2*J
          KL = J
   40   CONTINUE
        RETURN
        END

        SUBROUTINE ORD2(M, B)
        DIMENSION L(15)
        EQUIVALENCE (L15,L(1)), (L14,L(2)), (L13,L(3)), (L12,L(4)),
       *    (L11,L(5)), (L10,L(6)), (L9,L(7)), (L8,L(8)), (L7,L(9)),
       *    (L6,L(10)), (L5,L(11)), (L4,L(12)), (L3,L(13)), (L2,L(14)),
       *    (L1,L(15))
        DOUBLE PRECISION T, B(2)
        N = 2**M
        L(1) = N
        DO 10 K=2,M
          L(K) = L(K-1)/2
   10   CONTINUE
        DO 20 K M,14
          L(K+1) = 2
   20   CONTINUE
        IJ = 2
        DO 40 J1=2,L1,2
        DO 40 J2=J1,L2,L1
        DO 40 J3=J2,L3,L2
        DO 40 J4=J3,L4,L3
        DO 40 J5=J4,L5,L4
        DO 40 J6=J5,L6,L5
        DO 40 J7=J6,L7,L6
        DO 40 J8=J7,L8,L7
        DO 40 J9=J8,L9,L8
        DO 40 J10=J9,L10,L9
        DO 40 J11=J10,L11,L10
        DO 40 J12=J11,L12,L11
        DO 40 J13=J12,L13,L12
        DO 40 J14=J13,L14,L13
        DO 40 JI=J14,L15,L14
          IF (IJ-JI) 30, 40, 40
   30     T = B(IJ-1)
          B(IJ-1) = B(JI-1)
          B(JI-1) = T
          T = B(IJ)
          B(IJ) = B(JI)
          B(JI) = T
   40     IJ = IJ + 2
        RETURN
        END
```

A.4.4 : Error functions for NTF optimisation

For the optimisation of noise transfer functions an error criterion is required, independent of the optimisation routine used. Many different functions are required, each specific to the nature of the desired response. Typically, a balance must be struck between the gain of the NTF and the degree to which it fails to meet the noise stopband requirements. Since the stopband performance is architecture related, further constraints may be imposed such as some coefficients being zero-valued, some coefficients may be required to be integeric and some may be required for used in conventional IIR or FIR based noise shapers, others may be required for feedforward applications. Shown below is a typical error function for an NTF used in a noise shaper employing an IIR feedback filter, 'ERROR.FUN' (a similar function for an FIR filter used in a conventional design of noise shaper can be found in chapter four).

ERROR.FUN

```
      DOUBLE PRECISION FUNCTION ERROR(FAC)
c
c     NOZ = the number of delays in the iir (= (nop-1)/2 )
c       R = the current impulse response set (ntf : max 512)
c       T = the set of intermediate values within the iir
c
c     coefficients are read-in by the 'main' routine, assuming the direct
c     form II organisation :
c        k : the overall gain
c     fb(1) : the 1st feedback coefficient ("c" in the case of a Bi-Quad)
c     ff(1) : the 1st feedforward coeff't. ("b" in the case of a Bi Quad)
c     fb(2) : the 2nd feedback coefficient ("d" in the case of a Bi-Quad)
c     ff(2) : the 2nd feedforward coeff't. ("a" in the case of a Bi-Quad)
c     fb(3),ff(3), ... fb(noz)
c
      COMMON /EF/ NOP, NOS, C, N, NOB, DELTA, WEIGHT, R
      INTEGER NOB(514)
      DOUBLE PRECISION S, ERR, A(514), FAC(50), C(20),
     *                 DELTA(514), WEIGHT(514), R(514), T(11)
c
      DO 1 I=1,NOP                          ! find current coeffs
         A(I)=FAC(I)*C(I)
         IF (((A(I)**2-1.999)*(I/2-(I-1)/2)).GT.0.) THEN
             FAC(I)=FAC(I)/(A(I)/2.+0.005)**2   ! stability fail
             A(I)=FAC(I)*C(I)
         ENDIF
    1 CONTINUE
c     WRITE (0,'(5E12.4)') (A(I),I=1,NOP)
c
      NOZ=(NOP-1)/2                         ! find NTF by 1-IR
      R(1)=1.
      R(2)=-A(1)
      DO 2 I=2,NOZ                          ! null internal states
         T(I)=0.
    2 CONTINUE
      T(1)=A(1)
      ERR=1.+A(1)*A(1)
      DO 4 I=3,512
         T(11)=0.                           ! null accumulators
         R(I)=0.
         DO 3 K=NOZ,1,-1
            T(11)=T(11)+A(2*K)*T(K)         ! accumulate IIR section
            R(I)=R(I)+A(2*K+1)*T(K)         ! accumulate FIR section
            T(K)=T(K-1)                     ! clock delay contents ! undef 4 T(0)
    3    CONTINUE
         T(1)=T(11)
         R(I)=-(T(1)+R(I))
         ERR=ERR+R(I)*R(I)                  ! decompensate frequency domain
    4 CONTINUE
      R(513)=0.
      R(514)=0.
      FAC(49)=ERR                           ! sum of the squared sampled -IR
      CALL FFT(R,N)
      DO 6 I=1,NOS
         S=CDABS(DCMPLX(R(2*NOB(I)-1),R(2*NOB(I))))  ! MS FORTRAN COMMANDS.
         IF (S-DELTA(I)) 6,6,5
    5    ERR=ERR+WEIGHT(I)*((S/DELTA(I))**2-1.0D+0)  ! normalise before **2
    6 CONTINUE
      FAC(50)=ERR-FAC(49)  ! sum of the weighted squared errors of interest
      ERROR=ERR
      RETURN
      END
```

Appendix 5 : ASIC schematics & timing diagrams.

In the next three sections, schematic diagrams of circuits used in the thesis are presented in full. In this section, circuits designed using European Silicon Structures's design package 'SOLO' and Advanced Micro Devices' software 'PALASM' are presented. These were used for :

A.5.1 : the ASIC loading PAL presented in section 6.2.6,

A.5.2 : the design of the noise shaper ASIC,

A.5.3 : the assessment of the relative merits of fast counter structures,

and A.5.4 : the ASIC design (and testbed) for the PWM circuitry discussed in section 6.4.5.

The noise shaper design is split into two hierarchical sheets, the first at the top level the second with seven sub-blocks performing the limiter, quantiser, filter, input and output latches and the input summing node. A close up of the inside of one of the chips is also presented.

The fast counter structures are split over three sheets, the first showing four types of counter: a ripple counter based on J-K flip flips, a similar counter based on D-type flip-flops, a synchronous binary counter employing basic carry propagation circuitry, and a synchronous binary counter employing fast carry propagation circuitry, the second sheet showing the top level structure. The third sheet contains details of one implementation of a maximal length sequence counter and some ancillary functions for combining each of the test circuits into the test-bed.

The ASIC PWM schematics are spread over two sheets - the first showing the internal operation of the functional blocks 'load logic' and 'eight bit inverter' as used in the top level diagram : the second being the top level diagram itself.

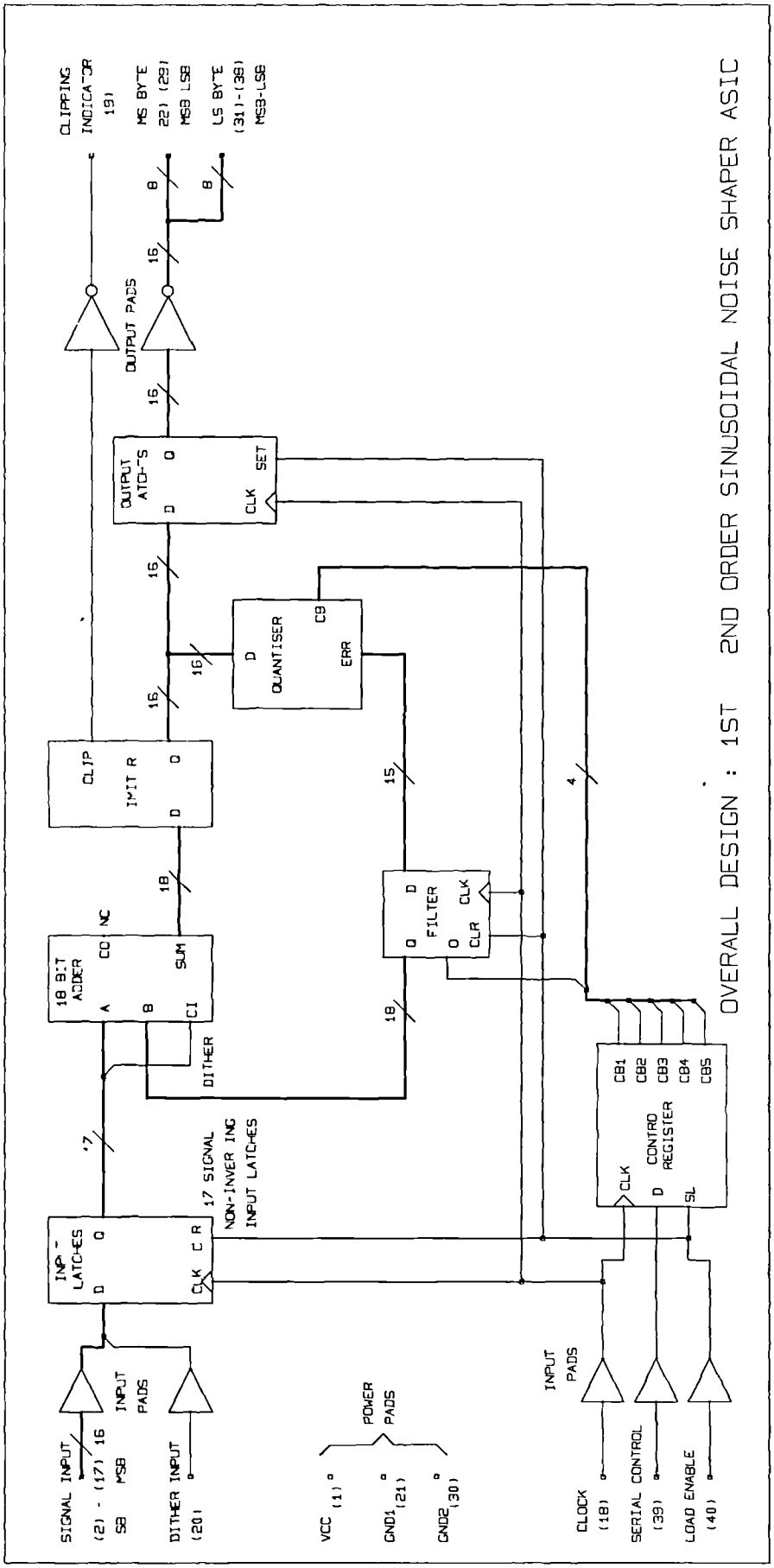
Shown below are the hardware description files for the programming and simulation of a 16R4 PAL with the ASIC control loading scheme described in section 6.2.6 . This PAL has been designed to provide simple parallel to serial conversion of five control bits CB1-5, along with support for a crystal oscillator circuit, complementary output, and two de-bounced AND-ed inputs.

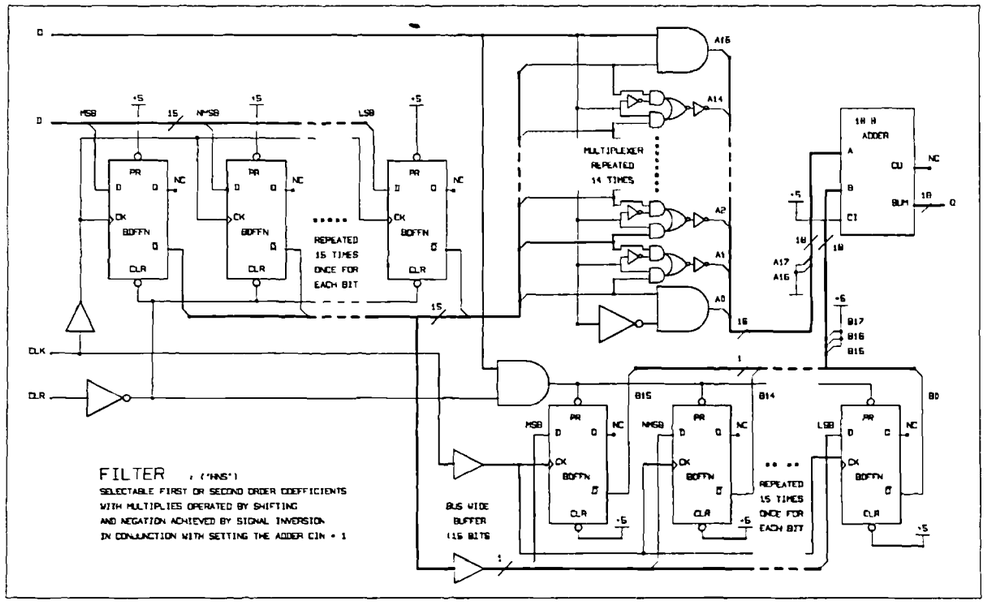
```
TITLE     ES2 ONS ASIC loading PAL
PATTERN   PTOSCON2.PDS
REVISION  2
AUTHOR    Rod. Hiorns.
COMPANY   British Technology Group
DATE      27/5/91

CHIP PTOSCON2 PAL16R4

clk cb1 cb2 cb3 cb4 cb5 xin   t2    t1 gnd
/oe /le /sd /q1 /q2 /q3 /q4 /nsd /xout vcc

EQUATIONS

q1 :=   /q3*/q2*(q1+t1+/t2)
q2 :=   /q3*(q2+q1)
q3 :=   (q2*/q1)
q4 :=   (q3*q1)

le   = /((q3+q2+q1)*/(q1*q3))
xout = xin
sd   = /(cb1*(/q3*/q2*q1) + cb2*(/q3*q2*q1) + cb3*(/q3*q2*/q1)
                          + cb4*(q3*q2*/q1) + cb5*(q3*/q2*/q1))

nsd  = cb1*(/q3*/q2*q1) + cb2*(/q3*q2*q1) + cb3*(/q3*q2*/q1)
                        + cb4*(q3*q2*/q1) + cb5*(q3*/q2*/q1)
```

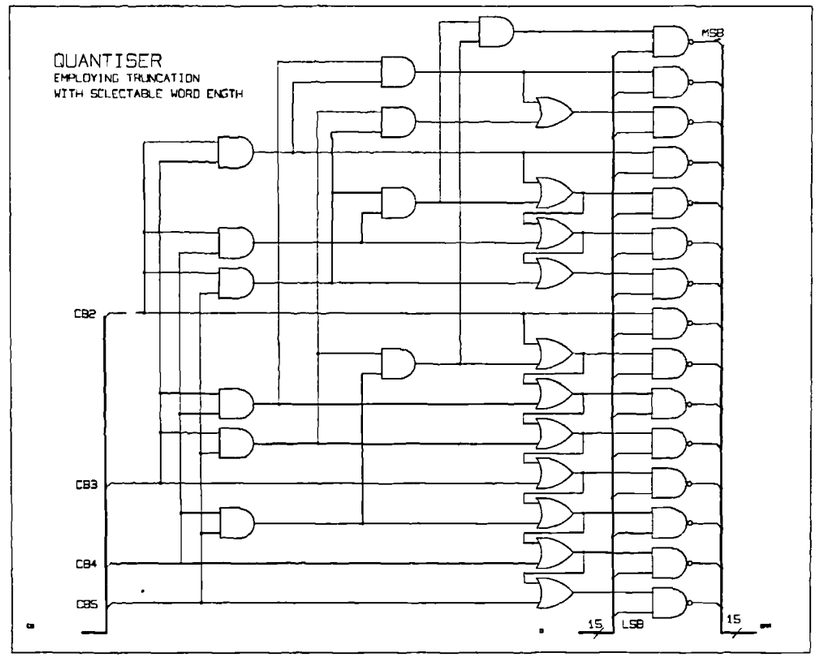
```
SIMULATION

TRACE_ON xin /xout cb1 cb2 cb3 cb4 cb5 t2 t1 clk /le /sd /nsd /oe /q1 /q2 /q3 /q4
SETF     /xin oe /clk /q1  /q2  /q3  /q4     ; reset o/p, check high Z
SETF     /oe /t1 t2 /cb1 /cb2 /cb3  cb4 /cb5 ; setup CB=00010, xin=1
CLOCKF   clk                                 ; clear power up state
SETF     oe                                  ; return active outputs
CLOCKF   clk                                 ; check wait state
SETF     xin t1                              ; xin=0, send CB: T=1
CLOCKF   clk                                 ; cb1 on /sd, /le=1
CLOCKF   clk                                 ; cb2 on /sd, /le=1
CLOCKF   clk                                 ; cb3 on /sd, /le=1
CLOCKF   clk                                 ; cb4 on /sd, /le=1
CLOCKF   clk                                 ; cb5 on /sd, /le=1
CLOCKF   clk                                 ; /sd=0, /le=0
SETF     /xin /t1 /t2 cb1 cb2 /cb3 /cb4 cb5  ; revise CB=11001, T=1
CLOCKF   clk                                 ; cb1 on /sd, /le=1
CLOCKF   clk                                 ; cb2 on /sd, /le=1
CLOCKF   clk                                 ; cb3 on /sd, /le=1
CLOCKF   clk                                 ; cb4 on /sd, /le=1
CLOCKF   clk                                 ; cb5 on /sd, /le=0
CLOCKF   clk                                 ; /sd=0, /le=0
SETF     t2                                  ; T=0
CLOCKF   clk                                 ; check stop state
TRACE_OFF
```

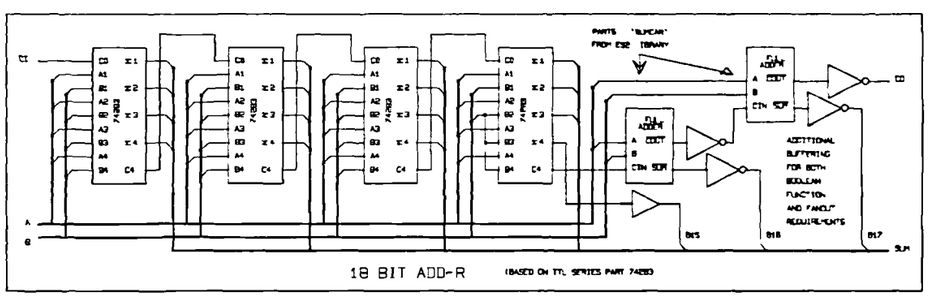
The resulting simulation output is as shown below:

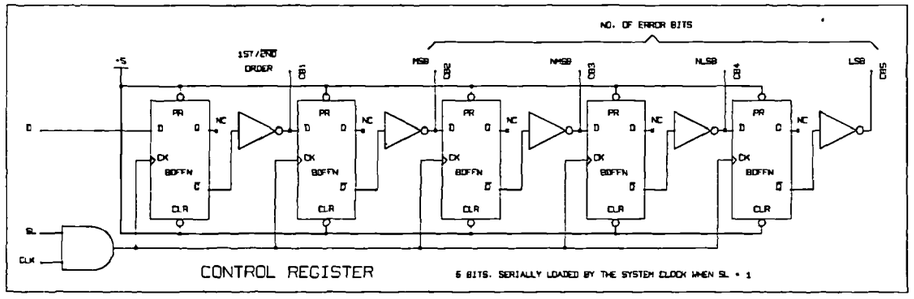
```
          ggc   gc   gc c   c   c   c   c    gc c   c   c   c   c   c    gc

  XIN   LLLLLLLLLHHHHHHHHHHHHHHHHHLLLLLLLLLLLLLLLLLLLLL
/XOUT   HHHHHHHHHLLLLLLLLLLLLLLLLLHHHHHHHHHHHHHHHHHHHH
  CB1   XLLLLLLLLLLLLLLLLLLLLLLLLLHHHHHHHHHHHHHHHHHHHH
  CB2   XLLLLLLLLLLLLLLLLLLLLLLLLLHHHHHHHHHHHHHHHHHHHH
  CB3   XLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
  CB4   XHHHHHHHHHHHHHHHHHHHHHHHHHLLLLLLLLLLLLLLLLLLLL
  CB5   XLLLLLLLLLLLLLLLLLLLLLLLLLHHHHHHHHHHHHHHHHHHHH
   T1   XHHHHHHHHHHHHHHHHHHHHHHHHHHLLLLLLLLLLLLLLLLHHHH
   T2   XLLLLLLLLHHHHHHHHHHHHHHHHHLLLLLLLLLLLLLLLLLLLL
  CLK   LLHHLLHHLLHHLHHLHHLHHLHHLLHHLHHLHHLHHLHHLLHHL
  /LE   LLLLLLLLLLLHHHHHHHHHHHHHHLLLLHHHHHHHHHHHHHLLLLL
  /SD   LLLLLLLLLLLLLLLLLLLLHHHLLLLLLLHHHHHHLLLLLLHHHLLLLL
 /NSD   HHHHHHHHHHHHHHHHHHHLLLHHHHHHLLLLLLHHHHHHLLLHHHH
  /OE   LHHHHLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
  /Q1   HZZZZHHHHHHLLLLLLHHHHHHHHHHHHLLLLLLHHHHHHHHHHH
  /Q2   HZZZZHHHHHHHHHLLLLLLLLLHHHHHHHHHHLLLLLLLLLHHHHHH
  /Q3   HZZZZHHHHHHHHHHHHHHLLLLLLHHHHHHHHHHHHLLLLLLHHHH
  /Q4   HZZZZHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
```

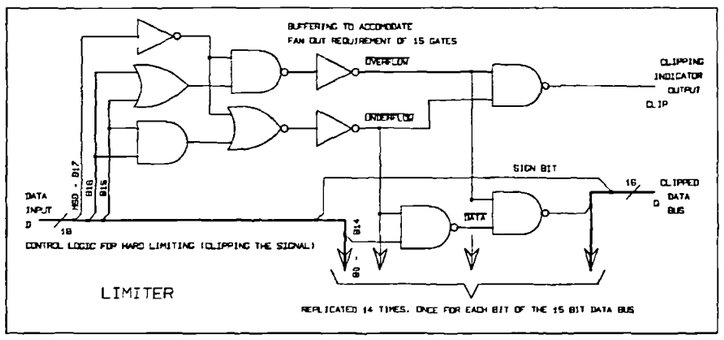

OVERALL DESIGN : 1ST 2ND ORDER SINUSOIDAL NOISE SHAPER ASIC
CLIPPING INDICATOR (19)
MS BYTE (22) (29) MSB LSB
LS BYTE (31)-(38) MSB-LSB
OUTPUT PADS
OUTPUT ATD-S
CLK SET
QUANTISER
CB
ERR
CLIP IMIT R
18 BIT ADDER
CO NC
SUM
CI
DITHER
FILTER
CLR
INPU- LATCHES
17 SIGNAL NON-INVER ING INPUT LATCHES
CB1 CB2 CB3 CB4 CB5
CONTRO REGISTER
SL
INPUT PADS
SIGNAL INPUT (2) - (17) 16 SB MSB
DITHER INPUT (20)
POWER PADS
VCC (1)
GND1 (21)
GND2 (30)
CLOCK (18)
SERIAL CONTROL (39)
LOAD ENABLE (40)

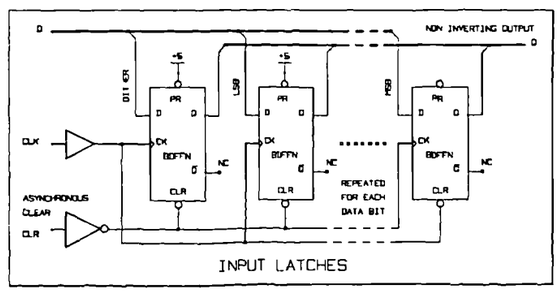

D
MSB
NMSB
LSB
+5
15
PR
CK
BDFFN
CLR
NC
REPEATED
15 TIMES
ONCE FOR
EACH BIT
CLK
CLR
MULTIPLEXER
REPEATED
14 TIMES
A16
A14
A2
A1
A0
16
1
18 H
ADDER
CU
BUM
18
+6
B17
B16
B15
B15
B14
B0
BUS WIDE
BUFFER
(15 BITS
REPEATED
15 TIMES
ONCE FOR
EACH BIT
FILTER
SELECTABLE FIRST OR SECOND ORDER COEFFICIENTS
WITH MULTIPLIES OPERATED BY SHIFTING
AND NEGATION ACHIEVED BY SIGNAL INVERSION
IN CONJUNCTION WITH SETTING THE ADDER CIN = 1

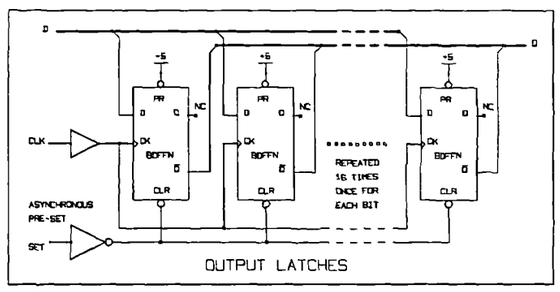

QUANTISER
EMPLOYING TRUNCATION
WITH SELECTABLE WORD ENGTH
MSB
CB2
CB3
CB4
CB5
15
LSB
15


CI
A
B
PARTS 'SUMCAR'
FROM ESE LIBRARY
ADDITIONAL
BUFFERING
FOR BOTH
BOOLEAN
FUNCTION
AND FANOUT
REQUIREMENTS
B15
B18
B17
SUM
18 BIT ADD-R
(BASED ON TTL SERIES PART 74283

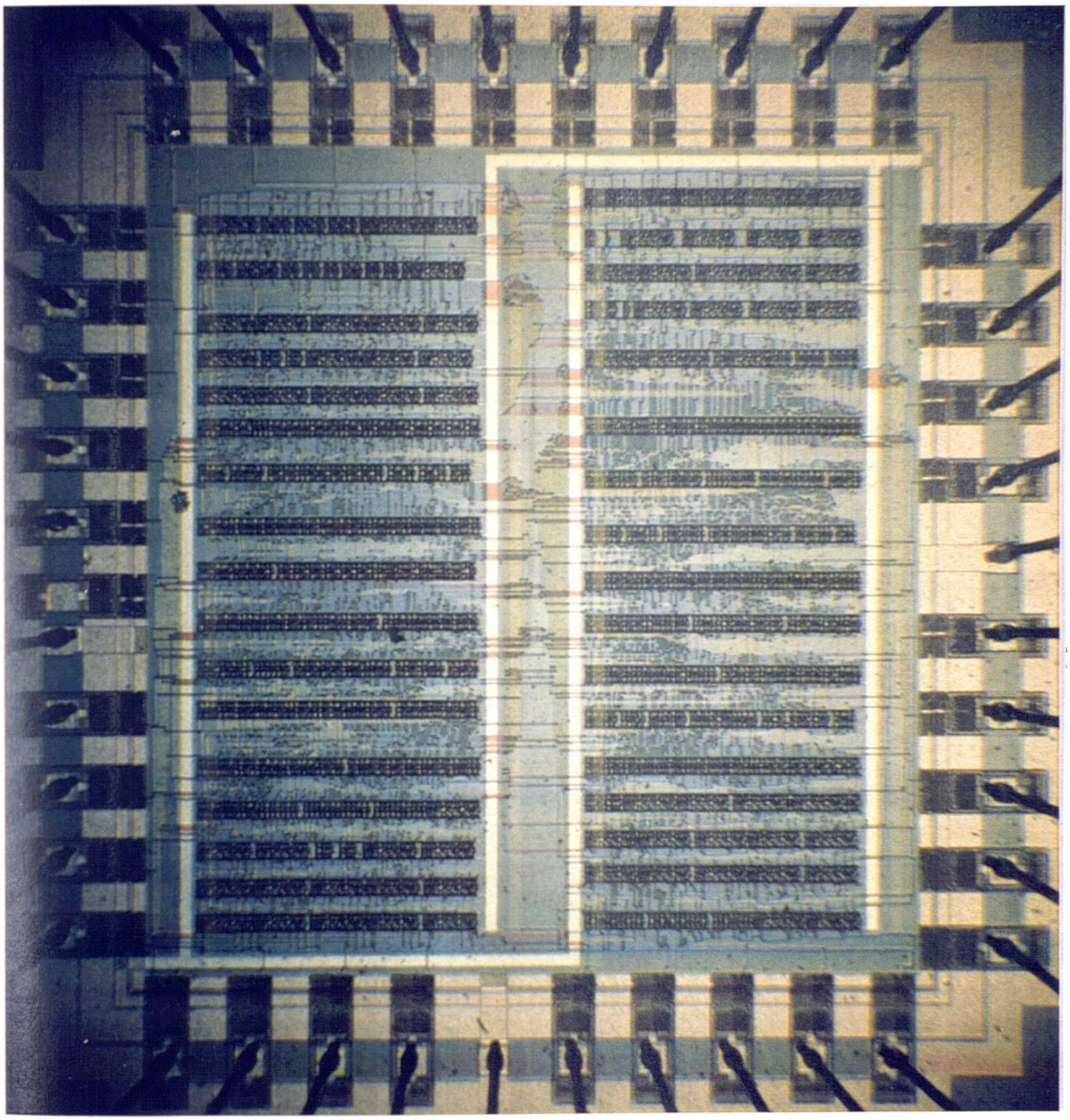

NO. OF ERROR BITS
1ST/2ND
ORDER
MSB
NMSB
NLSB
LSB
CB1
CB2
CB3
CB4
CB5
+5
D
PR
CK
BDFFN
CLR
NC
SL
CLK
CONTROL REGISTER
6 BITS. SERIALLY LOADED BY THE SYSTEM CLOCK WHEN SL = 1

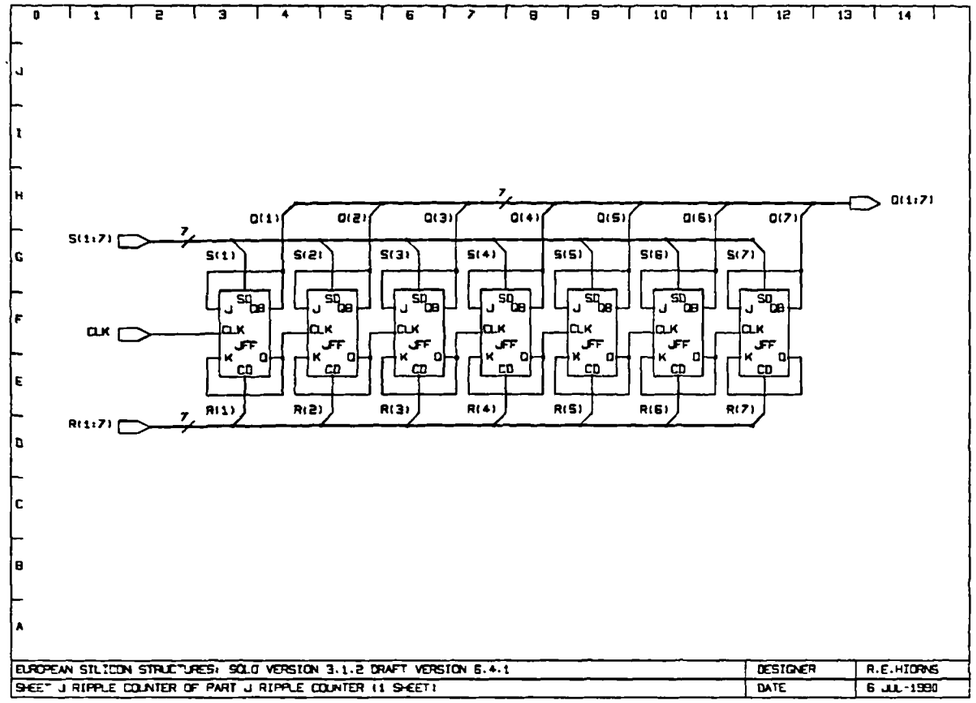

BUFFERING TO ACCOMODATE
FAN OUT REQUIREMENT OF 15 GATES
OVERFLOW
UNDERFLOW
CLIPPING
INDICATOR
OUTPUT
CLIP
SIGN BIT
16
CLIPPED
DATA
BUS
DATA
INPUT
D
18
MSB - B17
B16
B15
B14
B0
DATA
CONTROL LOGIC FOR HARD LIMITING (CLIPPING THE SIGNAL)
LIMITER
REPLICATED 14 TIMES. ONCE FOR EACH BIT OF THE 15 BIT DATA BUS


D
NON INVERTING OUTPUT
Q
+5
PR
CK
BDFFN
CLR
NC
LSB
MSB
CLK
ASYNCHRONOUS
CLEAR
CLR
REPEATED
FOR EACH
DATA BIT
INPUT LATCHES

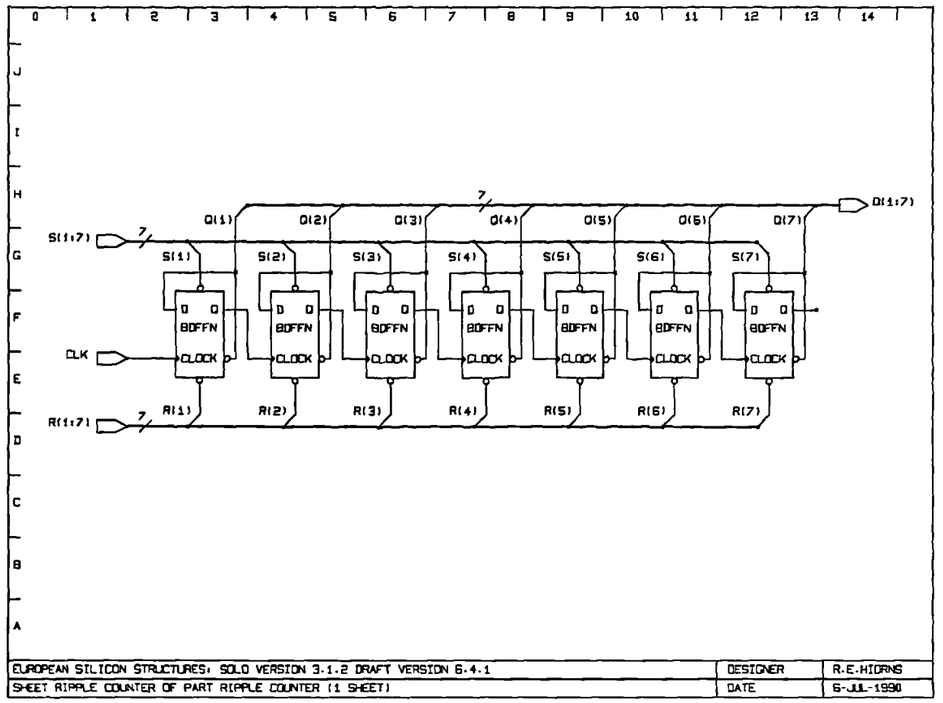

D
Q
+5
PR
CK
BDFFN
CLR
NC
CLK
ASYNCHRONOUS
PRE-SET
SET
REPEATED
15 TIMES
ONCE FOR
EACH BIT
OUTPUT LATCHES

The Internal View of the Noise Shaper ASIC
Viewed through a microscope (approx. x50)

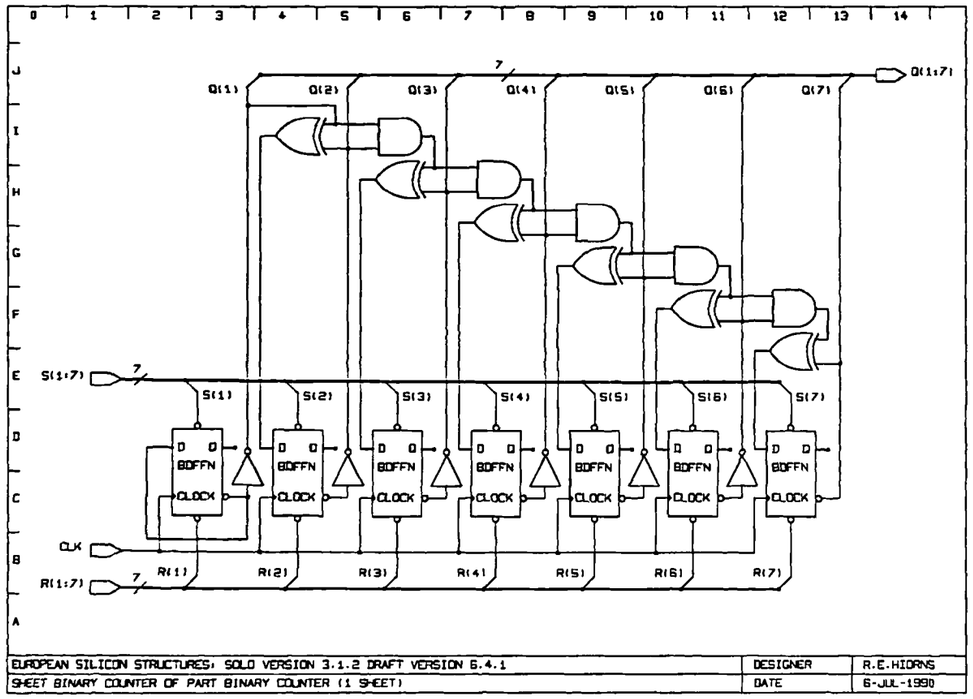




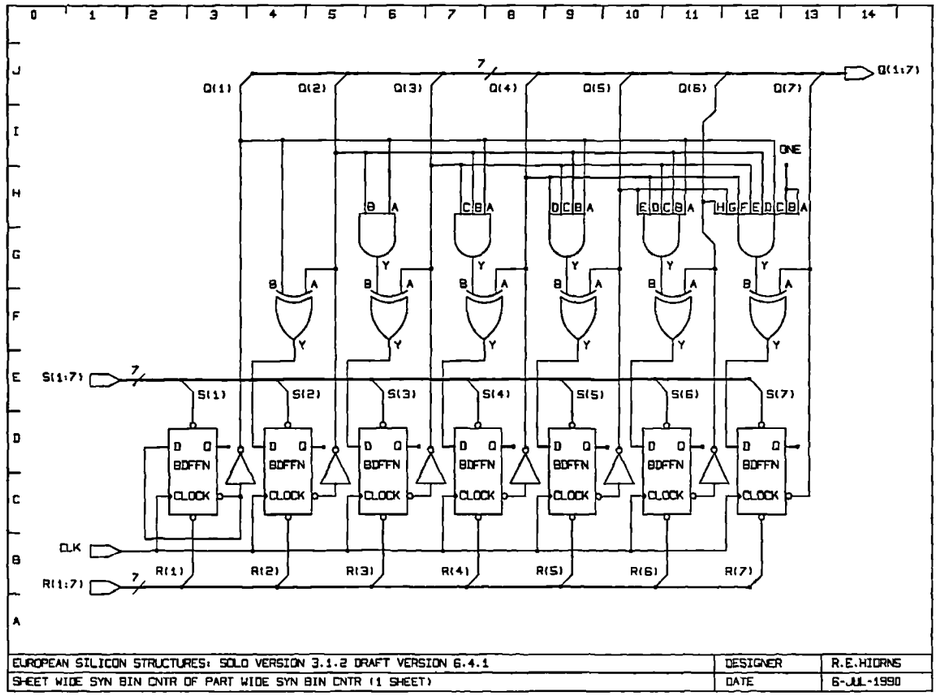
EUROPEAN SILICON STRUCTURES: SOLO VERSION 3.1.2 DRAFT VERSION 6.4.1	DESIGNER	R.E.HIORNS
SHEET J RIPPLE COUNTER OF PART J RIPPLE COUNTER (1 SHEET)	DATE	6 JUL-1990



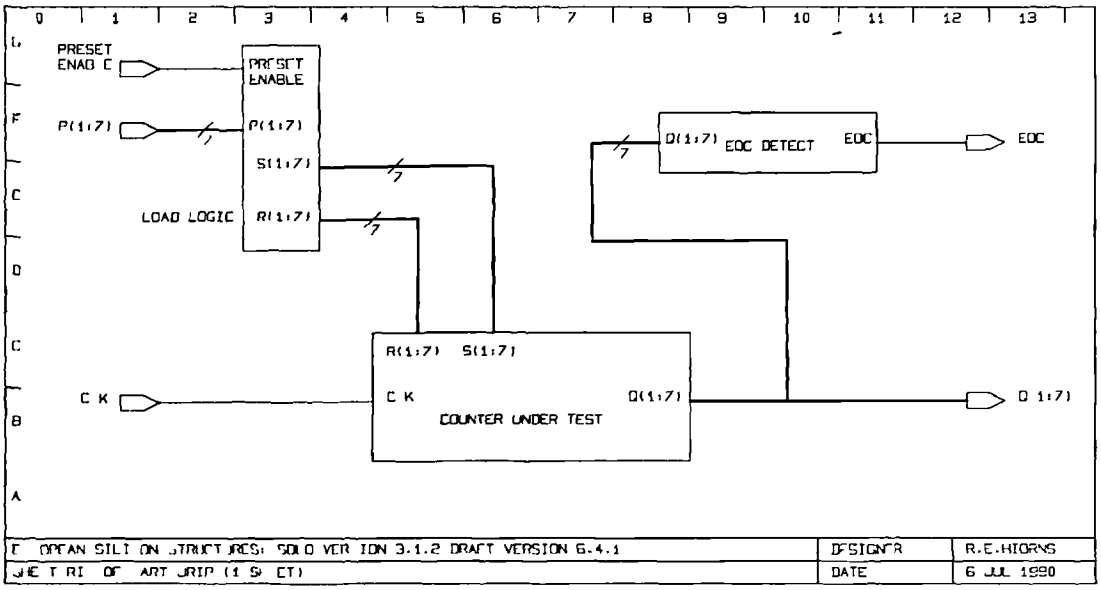
EUROPEAN SILICON STRUCTURES: SOLO VERSION 3.1.2 DRAFT VERSION 6.4.1	DESIGNER	R.E.HIORNS
SHEET RIPPLE COUNTER OF PART RIPPLE COUNTER (1 SHEET)	DATE	6-JUL-1990



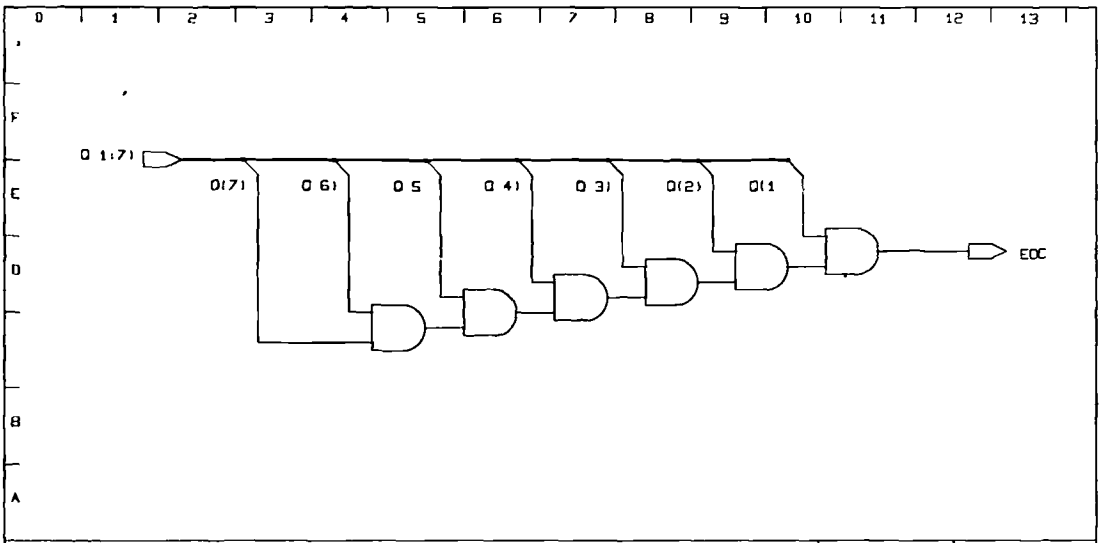
EUROPEAN SILICON STRUCTURES: SOLO VERSION 3.1.2 DRAFT VERSION 6.4.1	DESIGNER	R.E.HIORNS
SHEET BINARY COUNTER OF PART BINARY COUNTER (1 SHEET)	DATE	6-JUL-1990



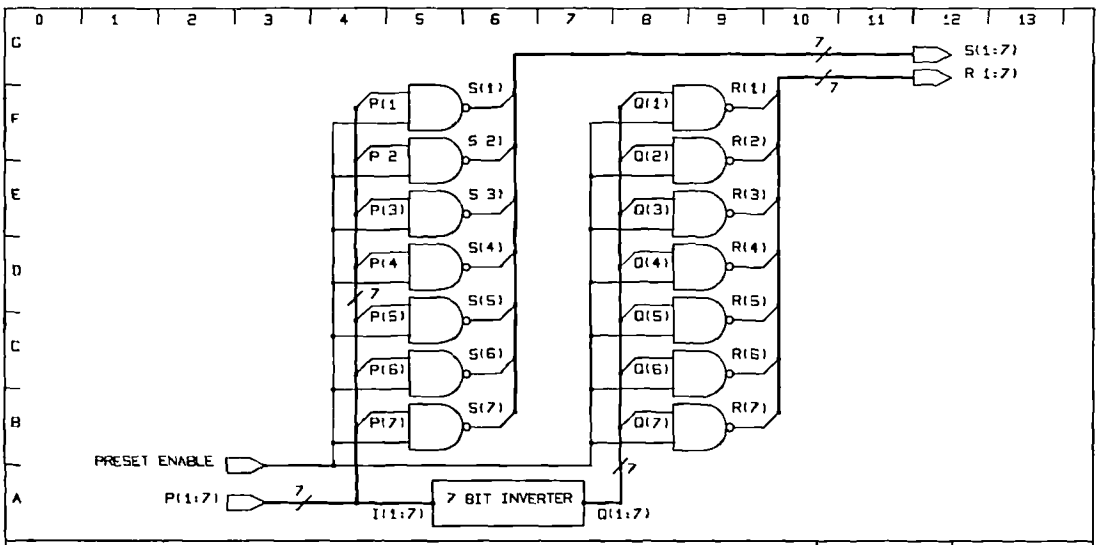
EUROPEAN SILICON STRUCTURES: SOLO VERSION 3.1.2 DRAFT VERSION 6.4.1	DESIGNER	R.E.HIORNS
SHEET WIDE SYN BIN CNTR OF PART WIDE SYN BIN CNTR (1 SHEET)	DATE	6-JUL-1990



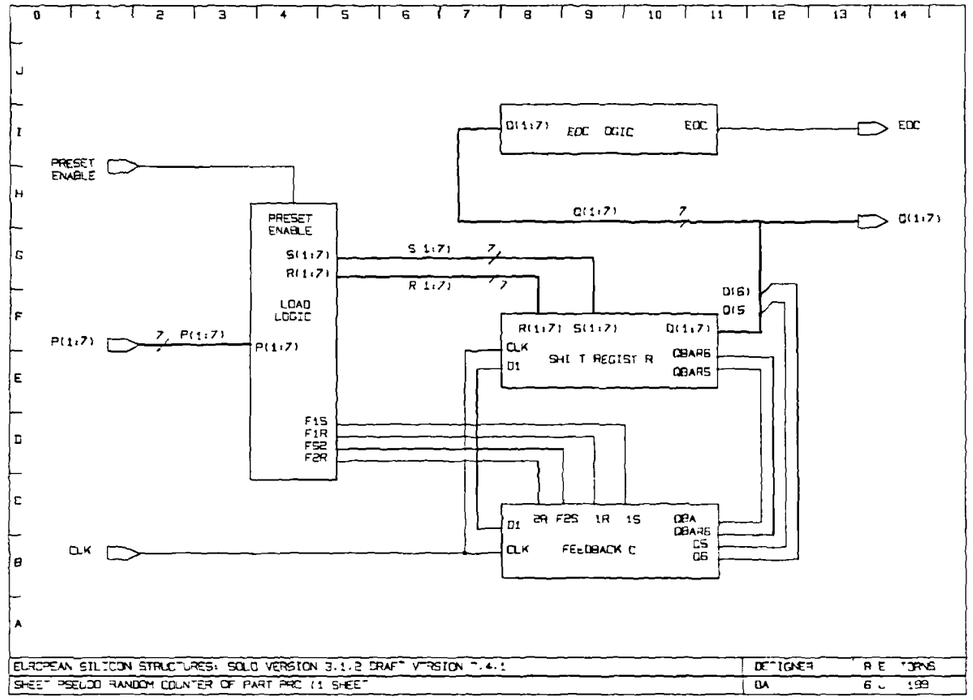
E OPEAN SILI ON STRUCTURES SOLO VER ION 3.1.2 DRAFT VERSION 6.4.1	DESIGNER	R.E.HIORNS
SHE T RI OF ART RIP (1 SHEET)	DATE	6 JUL 1990

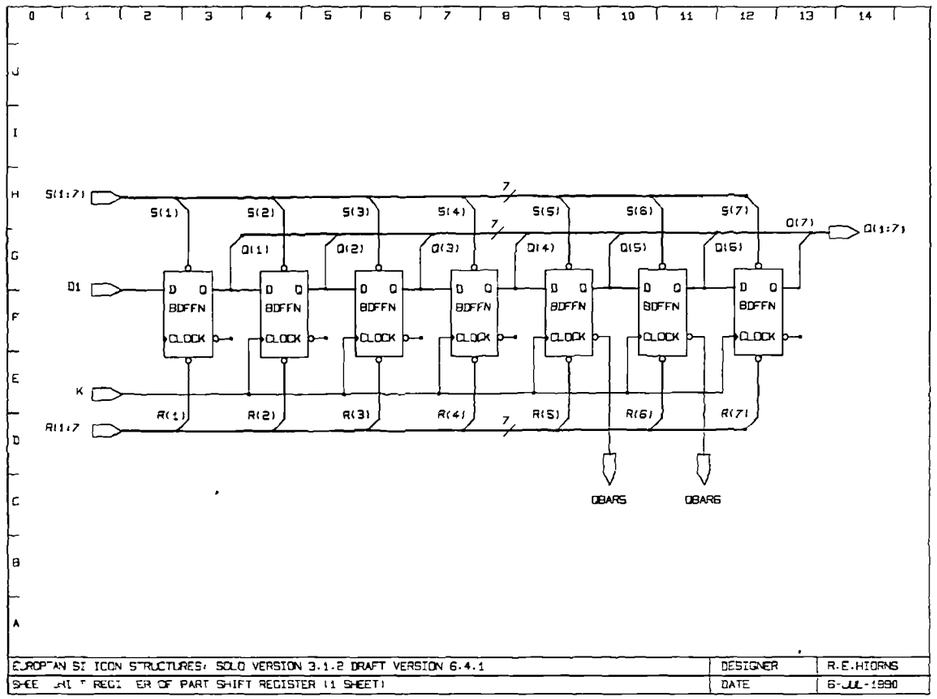


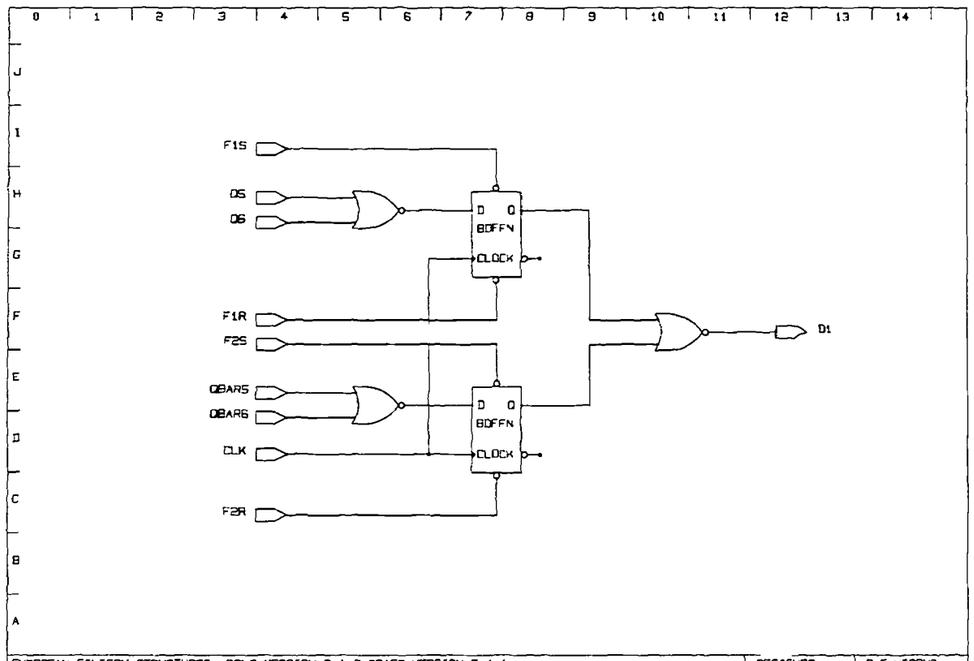
EUROPEAN SILI ON STRUCTURES. SOLO VERSION 3.1 2 DRAFT VERSION 6.4.1	DESIGNER	R.E.HIORNS
SHE T EOC DETECT OF PART EOC DETECT 1 SHEET)	DATE	6-JUL-1990

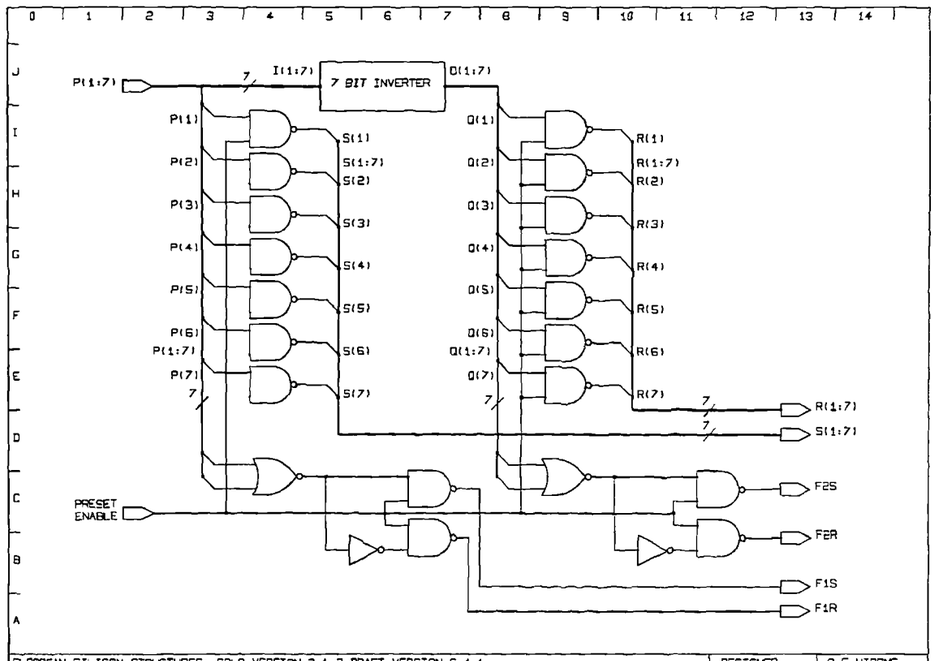


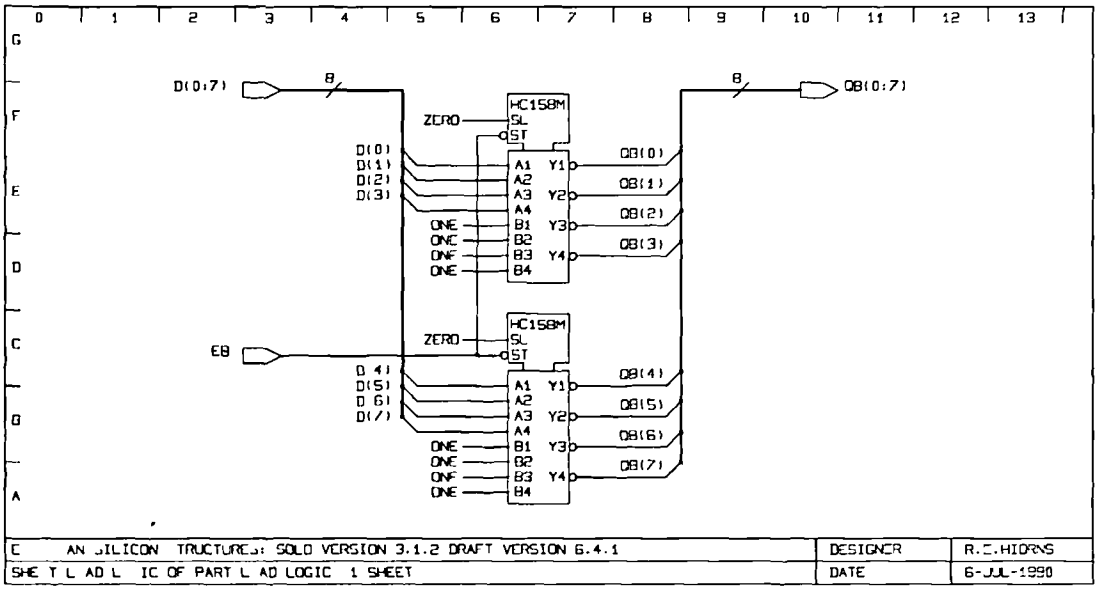
EUROPEAN SILICON STRUCTURES SOLO VERSION 3.1.2 DRAFT VERSION 6.4.1	DESIGNER	R.E.HIORNS
SH ET LOAD LOGIC OF PART LOAD LOGIC (1 SHEET)	DATE	6 JUL 1990

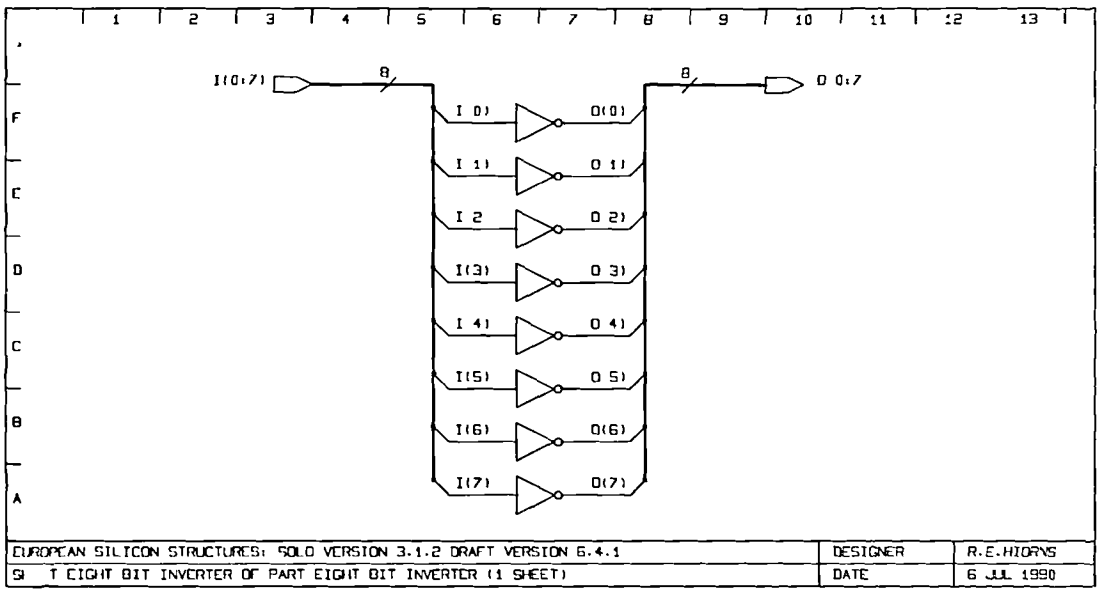

D(1:7) EOC OGIC
EOC
PRESET ENABLE
Q(1:7)
PRESET ENABLE
S(1:7)
R(1:7)
LOAD LOGIC
P(1:7)
P(1:7)
S 1:7)
R 1:7)
D(6)
D(5)
R(1:7) S(1:7) Q(1:7)
CLK
D1
SHI T REGIST R
QBAR6
QBAR5
F1S
F1R
F52
F2R
CLK
D1 2R F2S 1R 1S QBA
QBAR6
CLK FEcDBACK C
D5
D6

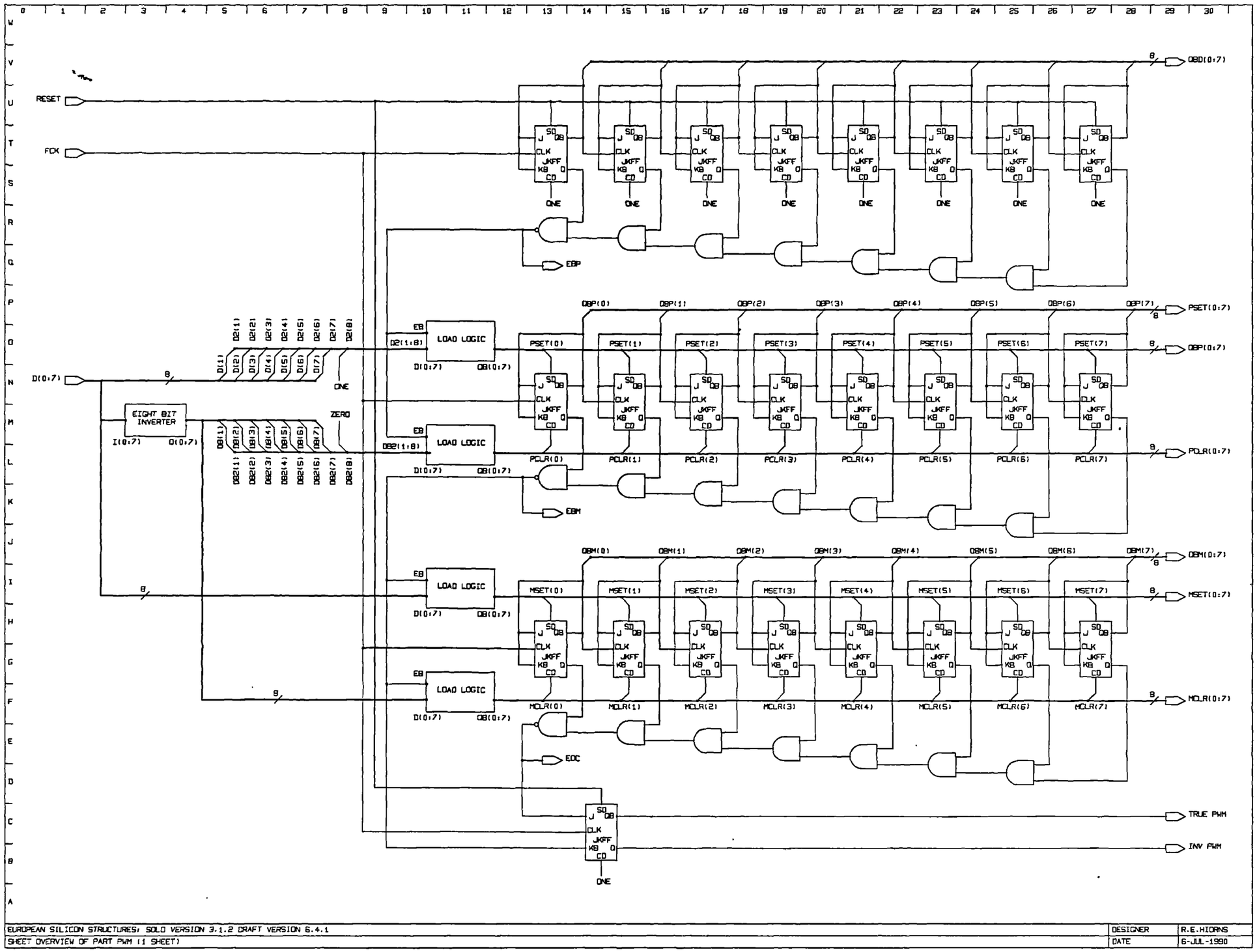

S(1:7)
7
S(1)
S(2)
S(3)
S(4)
S(5)
S(6)
S(7)
Q(7)
Q(1:7)
Q(1)
Q(2)
Q(3)
Q(4)
Q(5)
Q(6)
D1
D Q
BDFFN
CLOCK
K
R(1:7)
R(1)
R(2)
R(3)
R(4)
R(5)
R(6)
R(7)
QBAR5
QBAR6


F1S
D5
D6
D Q
BDFFN
CLOCK
F1R
F2S
QBAR5
QBAR6
CLK
F2R
D1


P(1:7)
I(1:7)
7 BIT INVERTER
D(1:7)
P(1)
P(2)
P(3)
P(4)
P(5)
P(6)
P(7)
P(1:7)
S(1)
S(1:7)
S(2)
S(3)
S(4)
S(5)
S(6)
S(7)
Q(1)
Q(2)
Q(3)
Q(4)
Q(5)
Q(6)
Q(7)
Q(1:7)
R(1)
R(1:7)
R(2)
R(3)
R(4)
R(5)
R(6)
R(7)
PRESET ENABLE
F2S
F2R
F1S
F1R


D(0:7)
8
HC158M
ZERO
SL
ST
D(0)
D(1)
D(2)
D(3)
A1 Y1
A2
A3 Y2
A4
ONE B1 Y3
ONE B2
ONE B3 Y4
ONE B4
8
DB(0:7)
DB(0)
DB(1)
DB(2)
DB(3)
EB
D(4)
D(5)
D(6)
D(7)
DB(4)
DB(5)
DB(6)
DB(7)
AN SILICON STRUCTURES SOLO VERSION 3.1.2 DRAFT VERSION 6.4.1
DESIGNER
R.E.HIORNS
DATE
6-JUL-1990


I(0:7)
8
I 0)
I 1)
I 2
I(3)
I 4)
I(5)
I(6)
I(7)
O(0)
O 1)
O 2)
O 3)
O 4)
O 5)
O(6)
O(7)
O 0:7
EUROPEAN SILICON STRUCTURES SOLO VERSION 3.1.2 DRAFT VERSION 6.4.1
DESIGNER
R.E.HIORNS
DATE
6 JUL 1990



EUROPEAN SILICON STRUCTURES, SOLO VERSION 3.1.2 DRAFT VERSION 6.4.1	DESIGNER	R.E.HIORNS
SHEET OVERVIEW OF PART PWM (1 SHEET)	DATE	6-JUL-1990

Appendix 6 : PCB Designs' Schematics.

In this appendix, five circuits used for experimenting with PWM are presented. They are organised in chronological order to follow the development of circuitry and ideas that has now led to practical PWM amplifier designs.

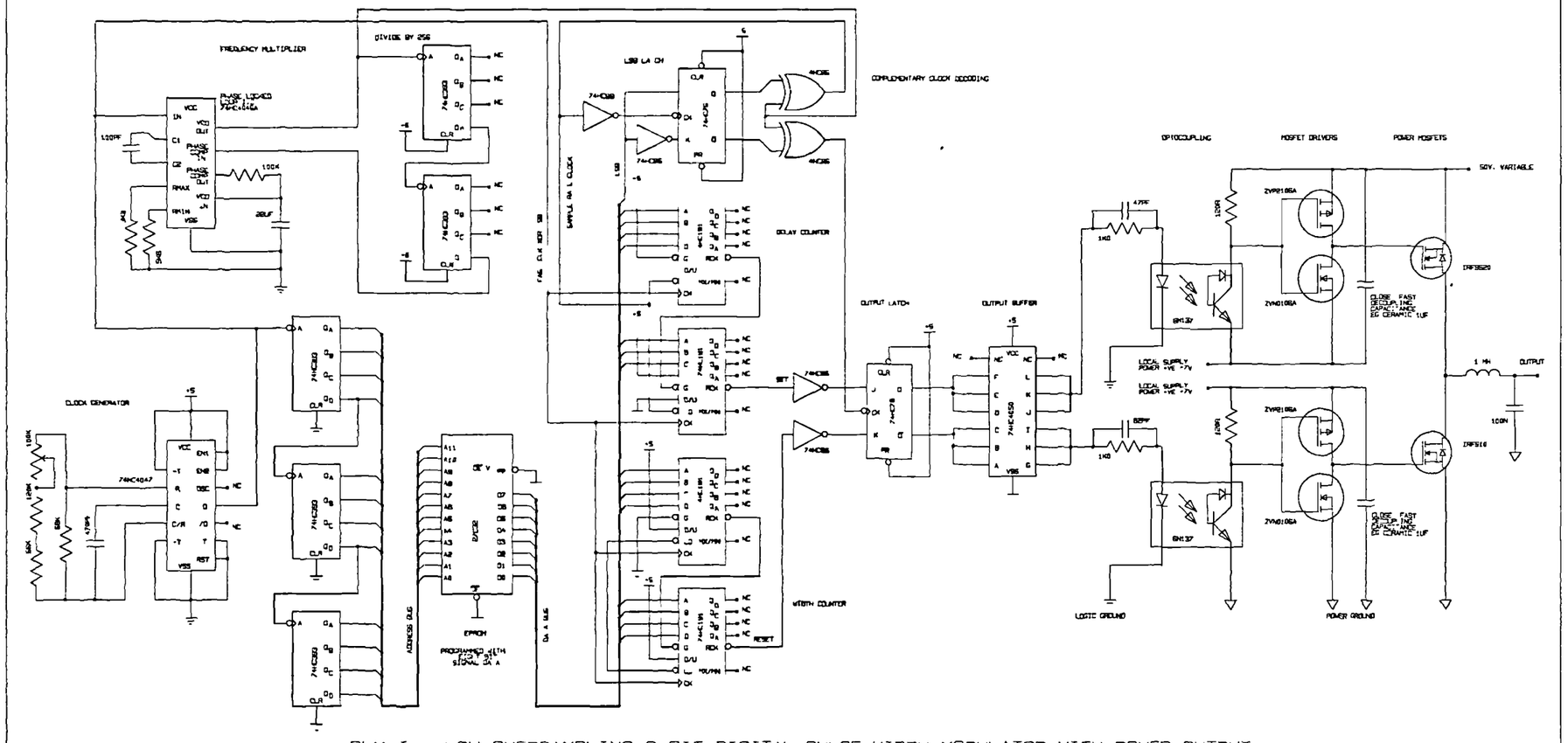
A.6.1 : PWM I is an 8 bit SYMPWM using alternating clock edges for fine resolution. It can be driven by a PLL multiplied clock sourced from an RC oscillator (15 MHz max.) and data stored in EPROM. The output can be amplified to 50v on-board.

A.6.2 : PWM II is a 12 bit modulator scanning RAM that can be programmed from a personal computer or the like. It can count at up to 144 MHz and produce TEPWM or SYMPWM from various wordlengths, oversampling and signal periods by appropriate configuration. The output is mono, complimentary and synchronous with an external clock.

A.6.3 : PWM III is a 9 bit two sided modulator, employing guard bands and capable of interfacing to some CD players. It has on-board noise shaping (up to 2nd order sinusoidal) and can be configured to produce almost any modulation type by appropriate use of its independent ports. It produces mono, complimentary output at TTL levels and up to 180 MHz.

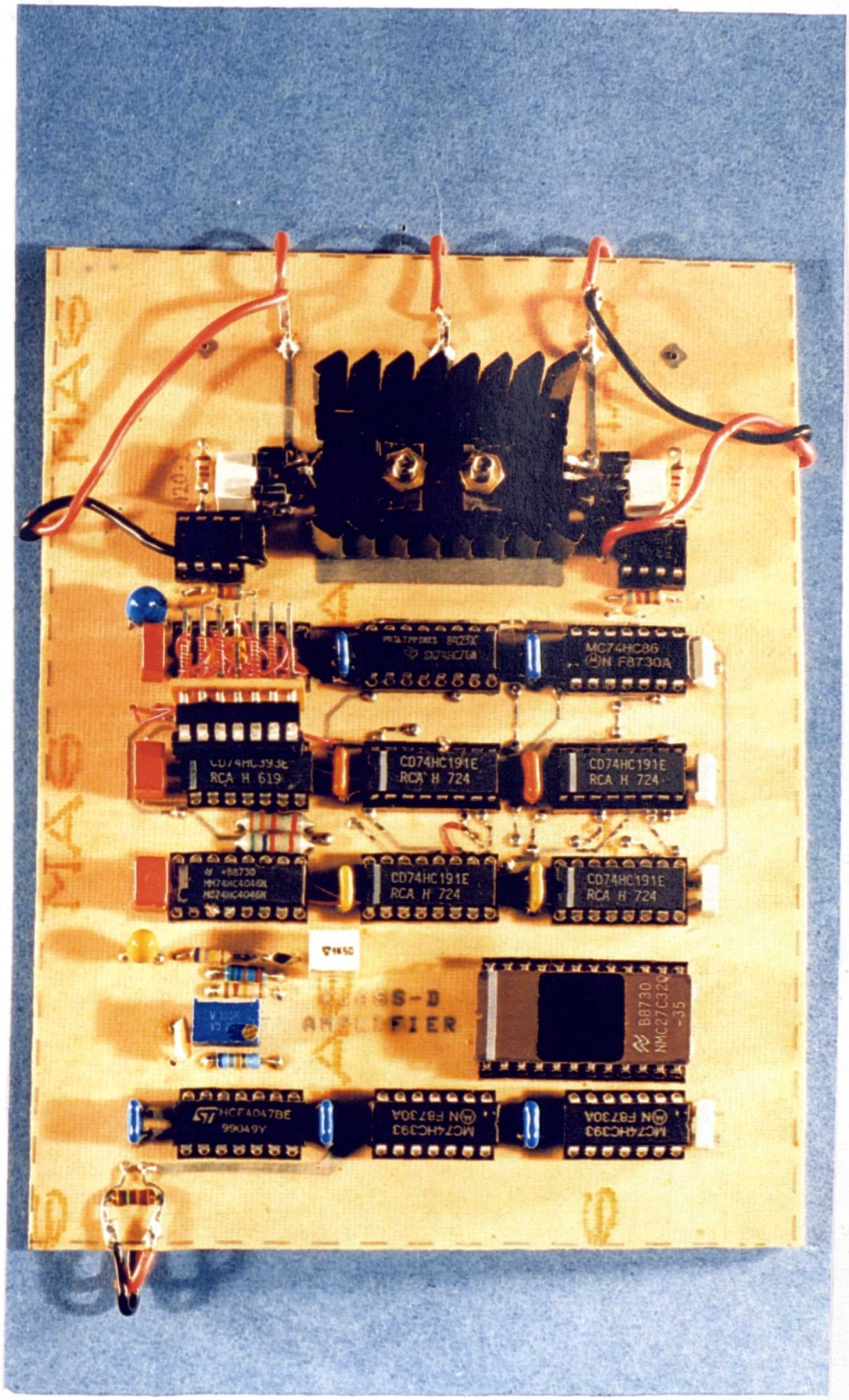
A.6.4 : PWM IV is similar to PWM 3 but has ECL output resynchronisation, PAL based counters (for the MSBs) and no noise shaper.

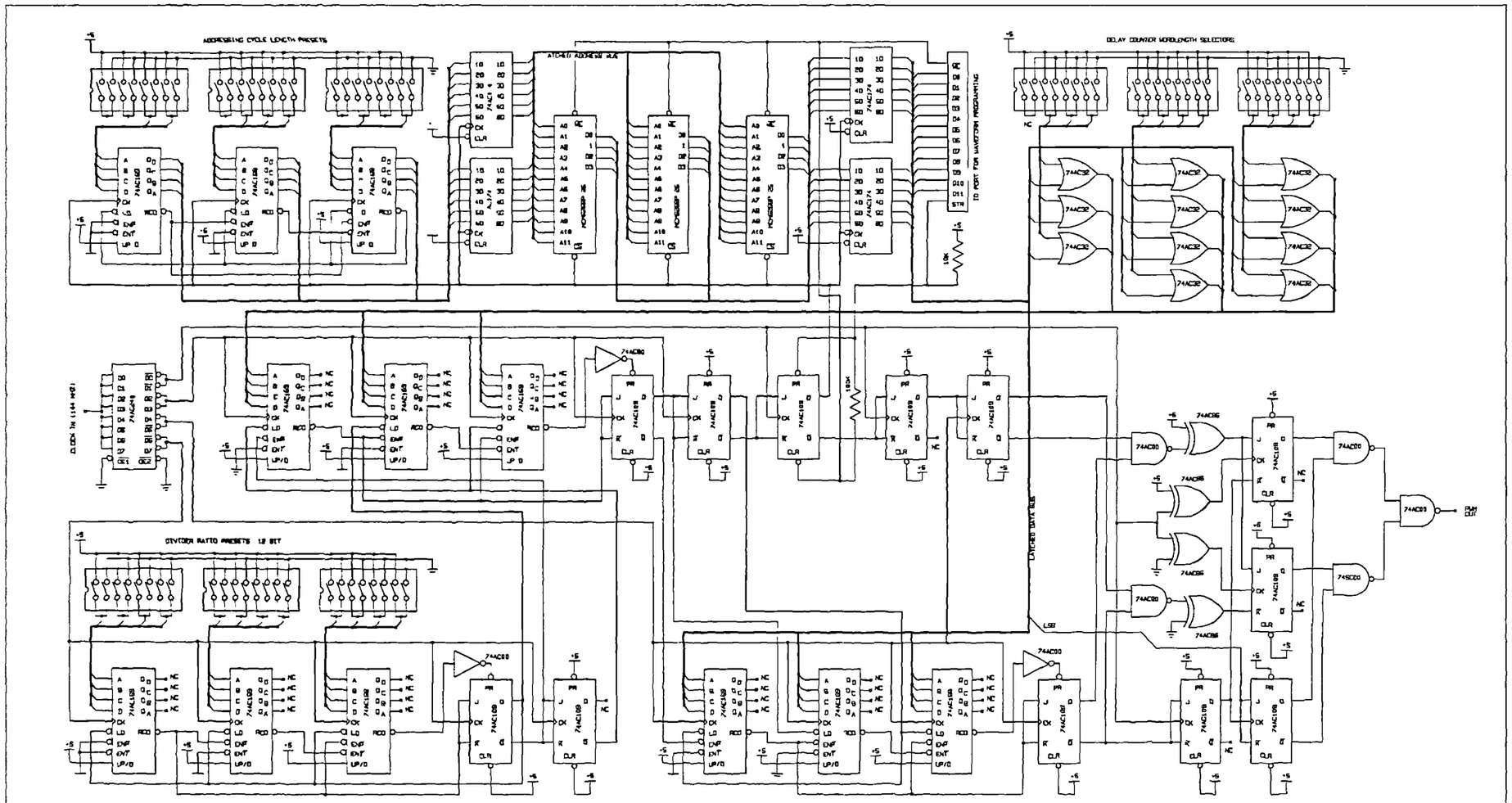
A.6.5 : PWM V interacts directly with a purpose built DSP board which when combined provide asynchronous SPDIF interface to audio input and stereo complementary PWM output at TTL levels. Its typical configuration would be for ESPWM and ZINS in assembly code on the DSP, using a PRR of 220 kHz and a counter clock of 126 MHz. It also includes additional power supply regulation and grounding routes to support low noise testing or interface.



PWM I : LOW OVERSAMPLING 8 BIT DIGITAL PULSE WIDTH MODULATOR WITH POWER OUTPUT
USING EPROM STORED SIGNAL DATA AND A FREQUENCY MULTIPLIED CLOCK SOURCE

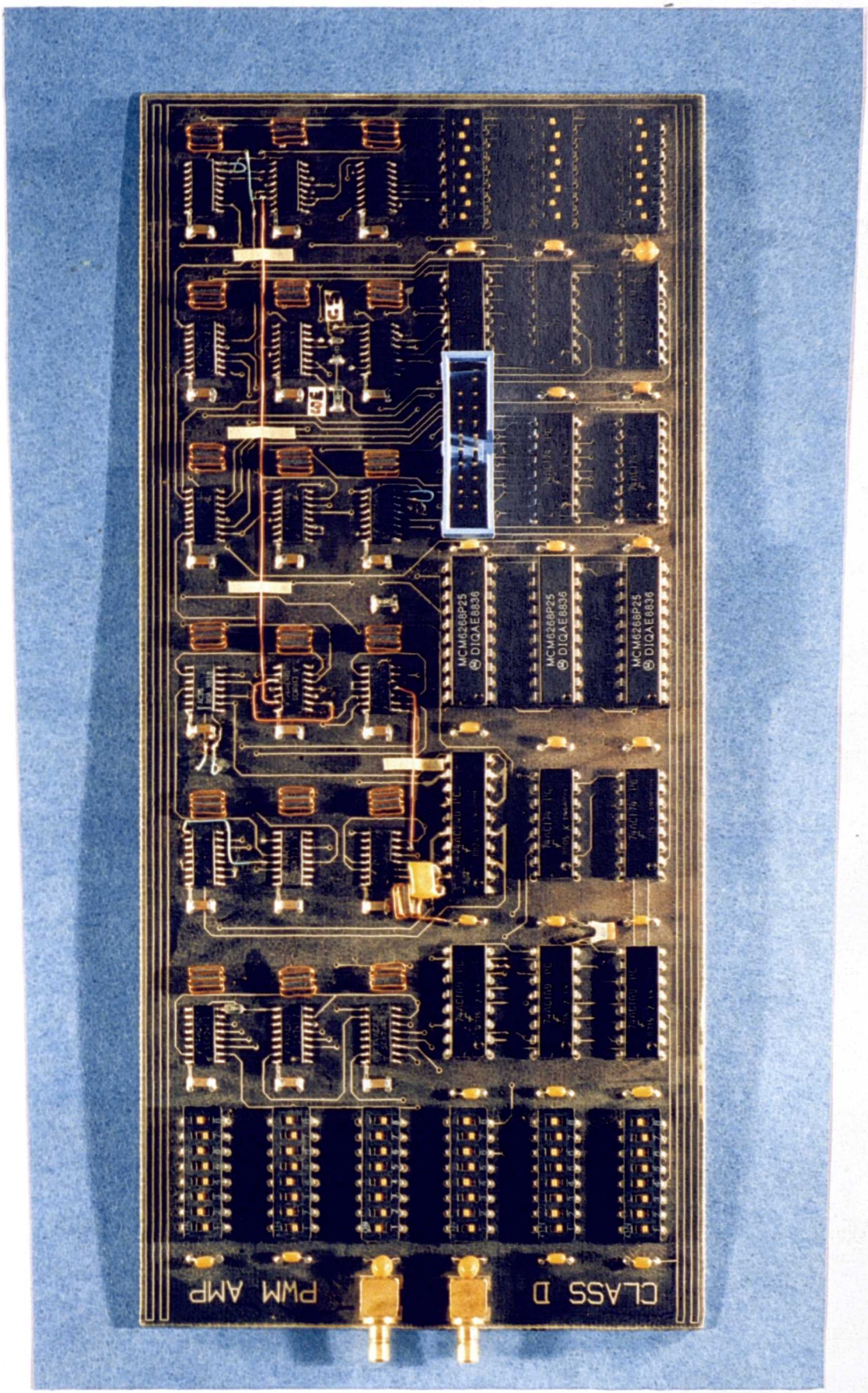
PCB Layout for PWM 1

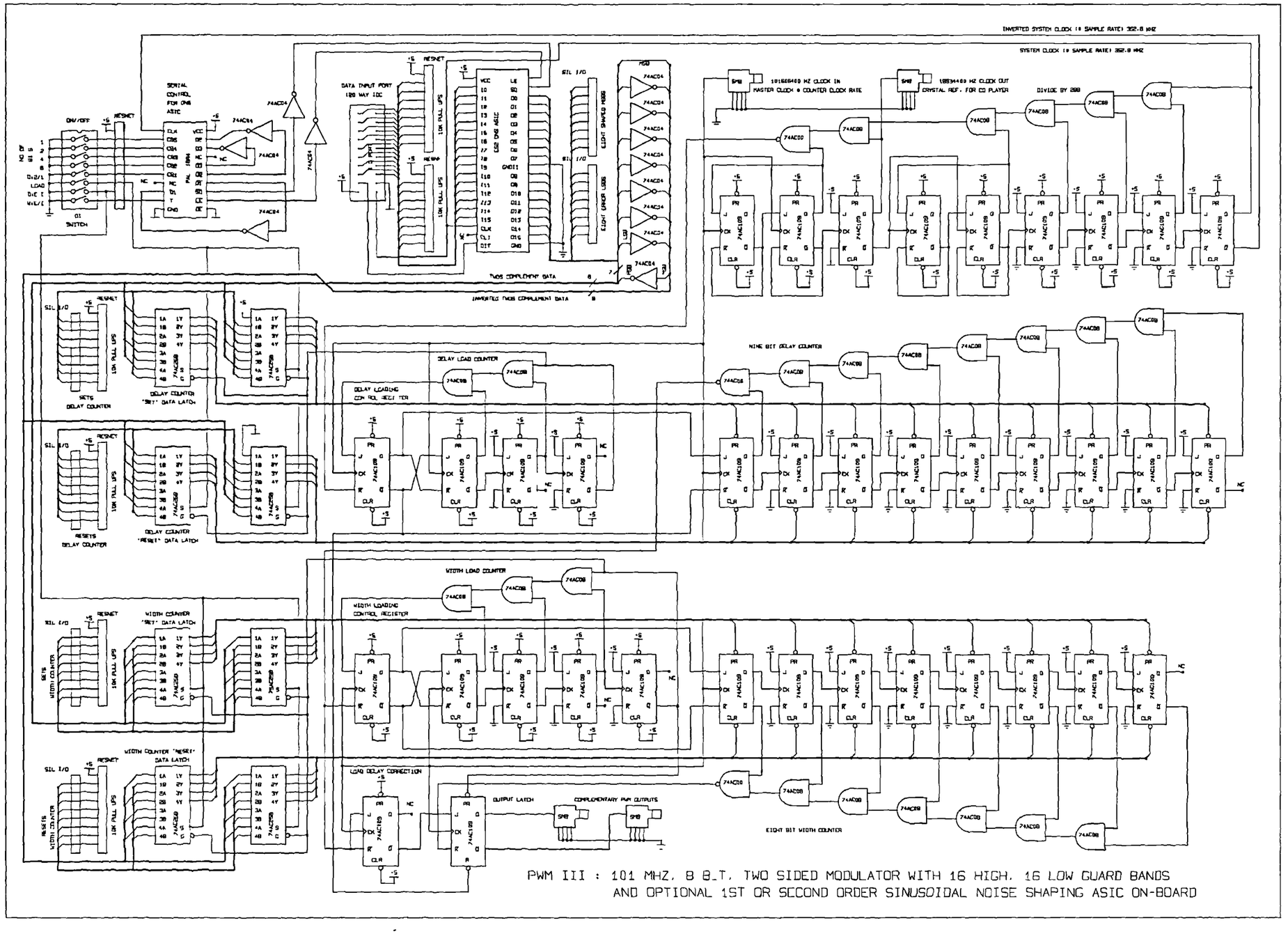




PWM II : HIGH SPEED (144 MHZ) PWM USING CYCLICALLY SCANNED MEMORY WAVEFORMS
CONFIGURABLE TO UPTO 12 BIT WIDTH RESOLUTION AND UPTO 2048X OVERSAMPLING

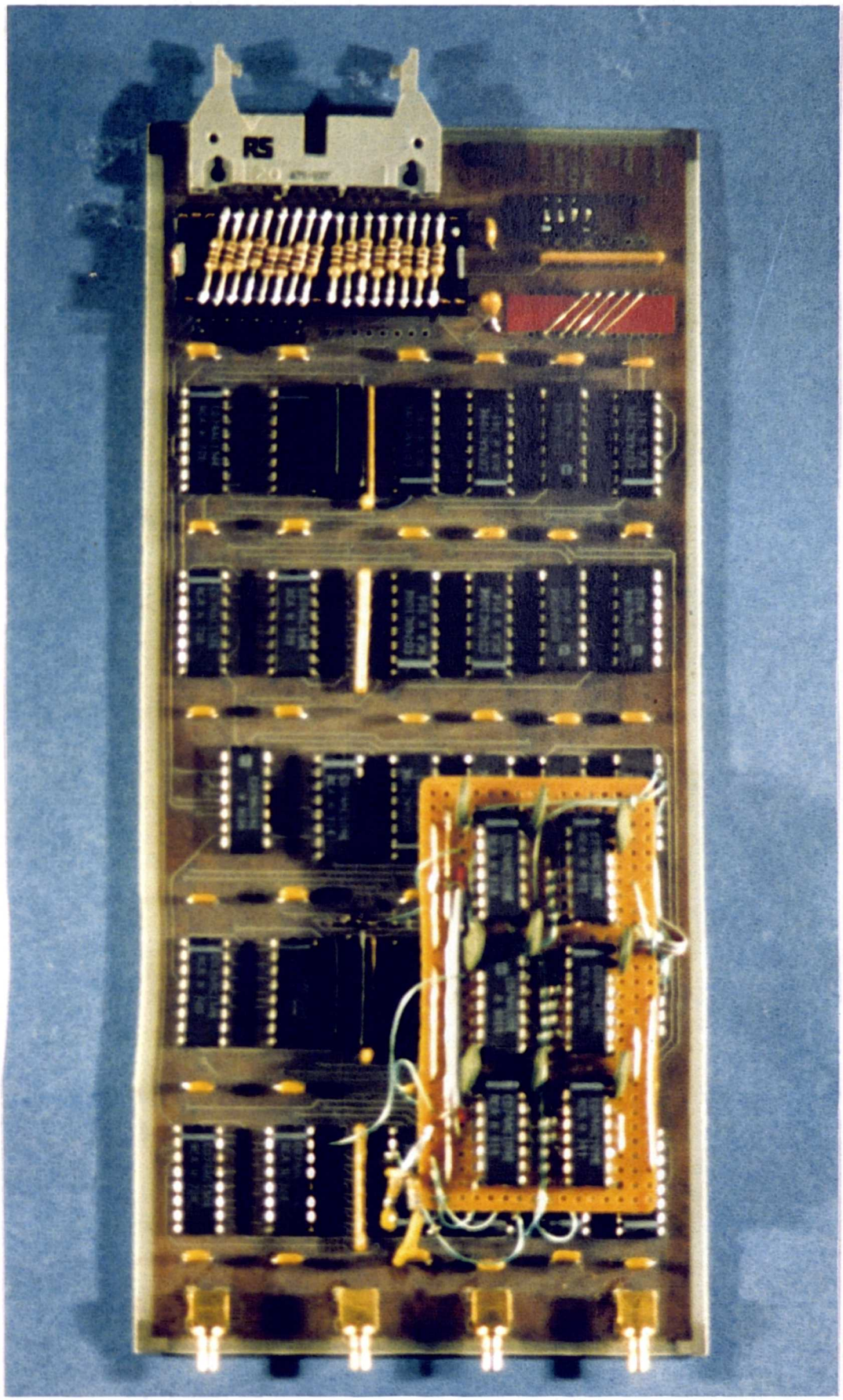
PCB Layout for PWM 2

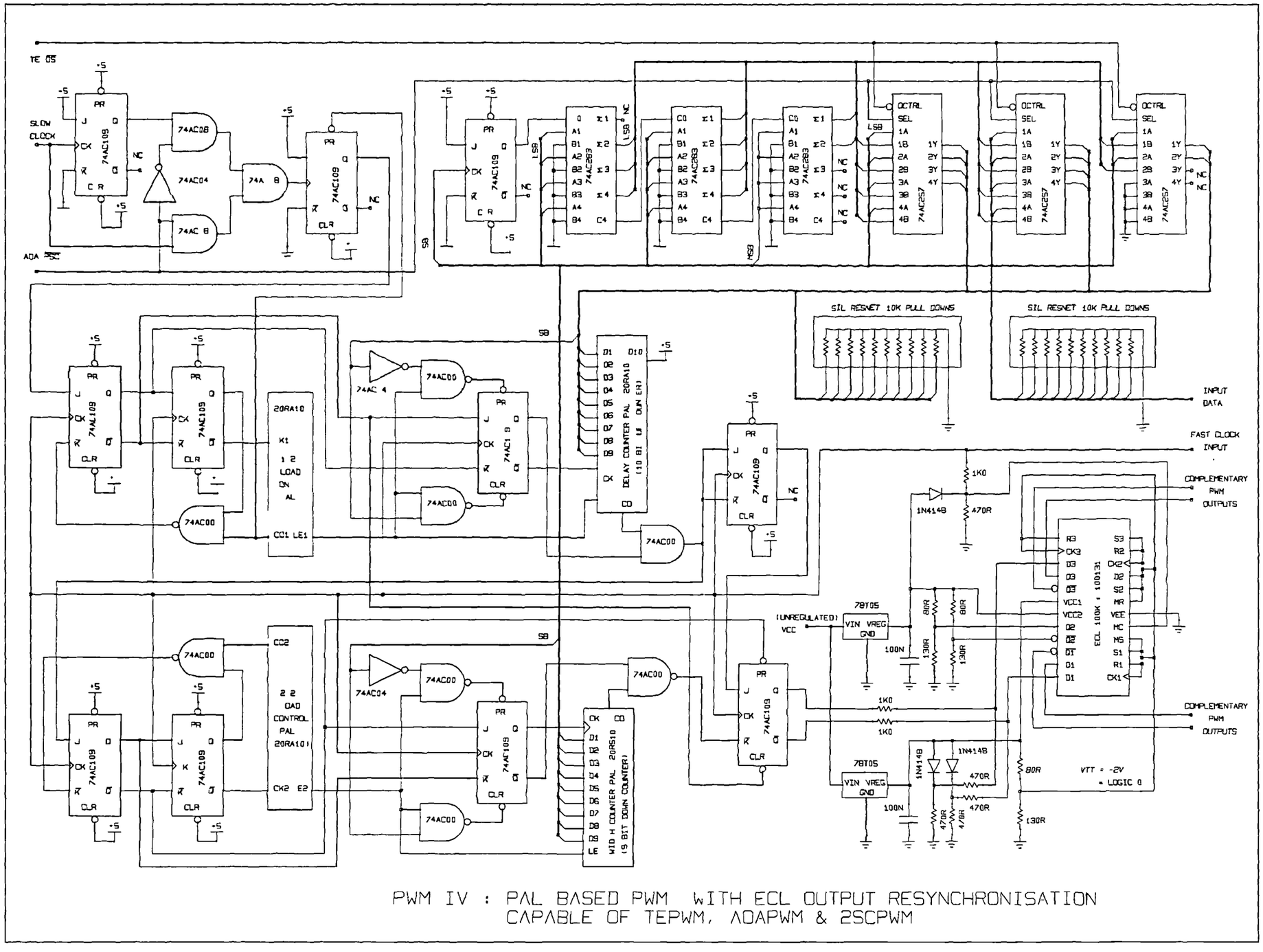




PWM III : 101 MHZ, 8 B_T, TWO SIDED MODULATOR WITH 16 HIGH, 16 LOW GUARD BANDS
AND OPTIONAL 1ST OR SECOND ORDER SINUSOIDAL NOISE SHAPING ASIC ON-BOARD

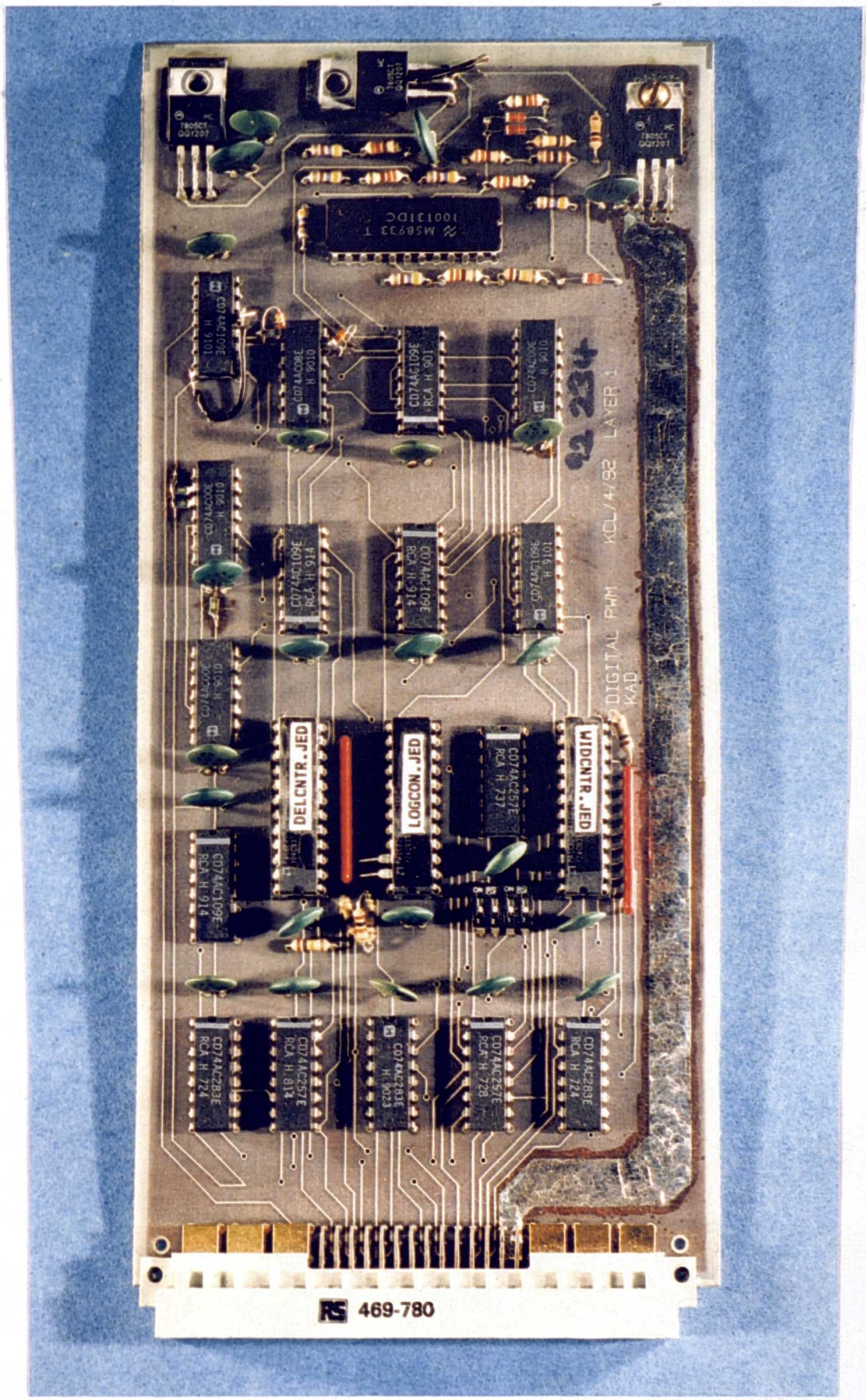
PCB Layout for PWM 3

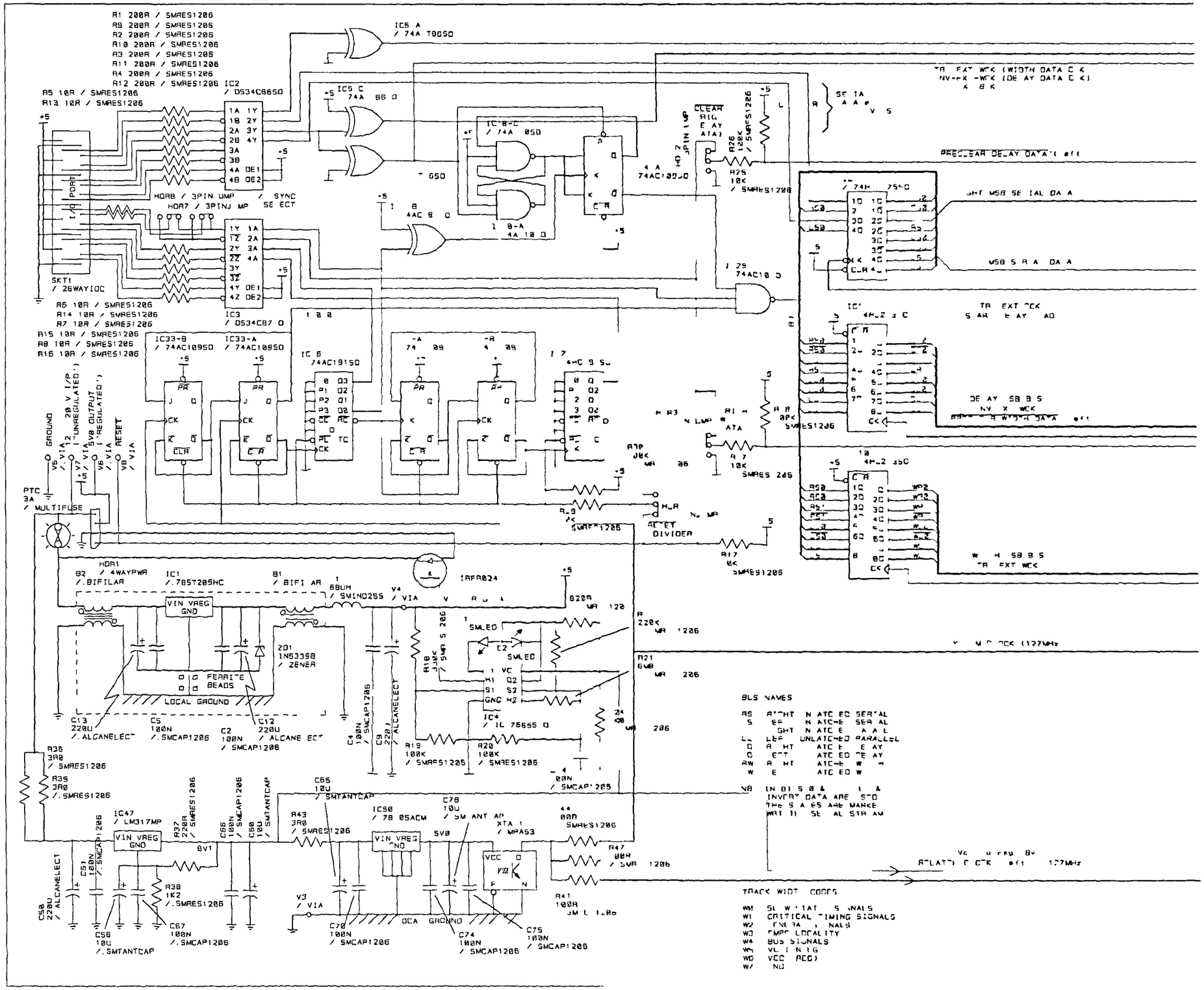




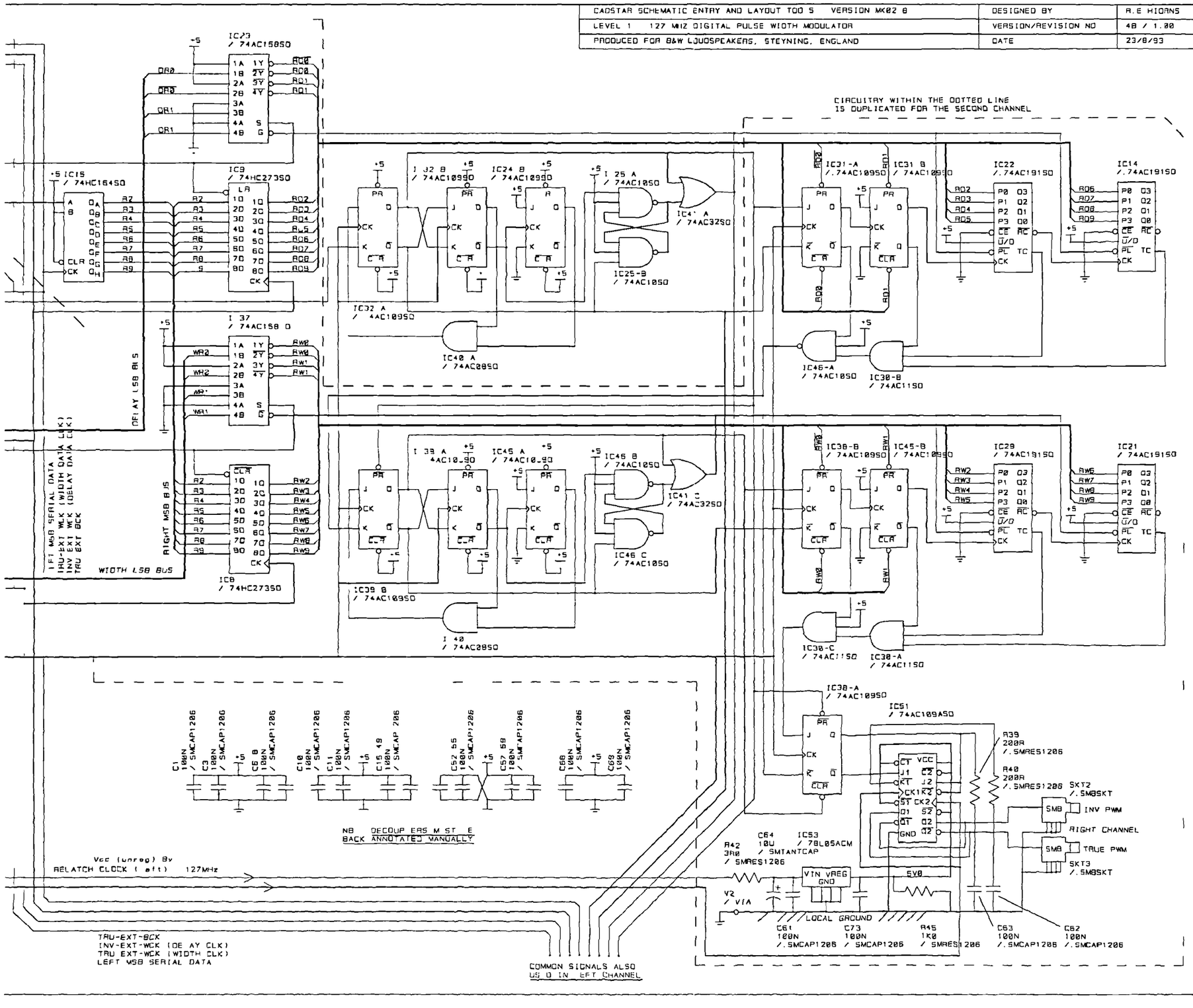
PWM IV : PAL BASED PWM WITH ECL OUTPUT RESYNCHRONISATION
CAPABLE OF TEPWM, AOAPWM & 2SCPWM

PCB Layout for PWM 4




R1 200R / SMRES1206
R9 200R / SMRES1206
R2 200R / SMRES1206
R10 200R / SMRES1206
R3 200R / SMRES1206
R11 200R / SMRES1206
R4 200R / SMRES1206
R12 200R / SMRES1206
R5 10R / SMRES1206
R13 10R / SMRES1206
IC2 / DS34C86SO
IC5 A / 74A T06SO
IC5 C 74A 86 O
IC8-C / 74A 08SO
IC8-A 4A 10 O
4 A 74AC10SO
CLEAR RIG E AY A(A)
R26 120K / SMRES1206
R25 10K / SMRES1206
PRECLEAR DE_AY DATA'(eff
SE IA A A e V S
TR FXT WCK (WIDTH DATA C K
NV-PX -WCK (DE AY DATA C K)
X 8 K
74H 75+0
WSB SE IAL DA A
WSB S R A DA A
I 25 74AC10 O
IC' 4HC2 3 C
TR EXT TCK
S AR E AY AO
DE AY SB B S
NV X WCK
R7 10K SMRES 206
R17 8K SMRES1206
HDR8 / 3PIN UMP / SYNC
HDR7 / 3PINJ MP SE ECT
SKT1 / 26WAYIOC
R6 10R / SMRES1206
R14 10R / SMRES1206
R7 10R / SMRES1206
IC3 / DS34C87 O
R15 10R / SMRES1206
R8 10R / SMRES1206
R16 10R / SMRES1206
IC33-B / 74AC109SO
IC33-A / 74AC109SO
IC 6 74AC191SO
74 09
4 09
4HC 9 SO
R39 10K
R43 3R0 / SMRES1206
IC50 / 78 05ACM
C78 10U / SM ANT AP
XTA 1 / MPA53
R47 00R / SMR 1206
R41 100R
C65 10U / SMTANTCAP
C66 100N / SMCAP1206
C60 10U / SMTANTCAP
R37 220R / SMRES1206
IC47 / LM317MP
R38 1K2 / SMRES1206
C51 100N / SMCAP1206
C58 220U / ALCANELECT
C56 10U / SMTANTCAP
C67 100N / SMCAP1206
C70 100N / SMCAP1206
C74 100N / SMCAP1206
C75 100N / SMCAP1206
R36 3R0 / SMRES1206
R35 3R0 / SMRES1206
PTC 3A / MULTIFUSE
GROUND
20 V I/P ("UNREGULATED")
5V0 OUTPUT ("REGULATED")
RESET
HDR1 / 4WAYPWR
IC1 / 785T205HC
B1 / BIFI AR
68UH / SWIND25S
IRFR024
VIN VREG GND
SMLED
C2 SMLED
ZD1 1N5339B / ZENER
FERRITE BEADS
LOCAL GROUND
C13 220U / ALCANELEC
C5 100N / SMCAP1206
C12 220U / ALCANE ECT
C4 100N / SMCAP1206
C9 220U / ALCANELECT
R18 330K / SMR S 206
R19 100K / SMRES1206
R20 100K / SMRES1206
IC4 / IL 7655S O
R21 6K8 SMR 206
BUS NAMES
TRACK WIDT CODES
W0 SE W IAT S NALS
W1 CRITICAL TIMING SIGNALS
W2 ENERA NALS
W3 SMPC LOCALITY
W4 BUS SIGNALS
W5 VE I NIG
W6 VCC REG)
W7 NU

CADSTAR SCHEMATIC ENTRY AND LAYOUT TOO S VERSION MK02 8	DESIGNED BY	R.E HIORNS
LEVEL 1 127 MHZ DIGITAL PULSE WIDTH MODULATOR	VERSION/REVISION NO	48 / 1.00
PRODUCED FOR B&W LOUDSPEAKERS, STEYNING, ENGLAND	DATE	23/8/93



Appendix 7 : Prototyped Designs' Schematics.

In this appendix, prototyped circuits are presented which may not be as large or complex as the PWMs or noise shaper mentioned earlier, but do however present a convenient solution to problems encountered while developing PWMs.

Common areas of concern are the generation of a high quality clock source to define the fast counter clock in the DPWM itself (and possibly to provide synchronisation signals after dividing down), how to keep the output operating as a voltage source even while supplying current, how to prevent conducted interference (especially mains hum and television 'flyback' signals).

To add to this, one example of a simple high speed pre-processing circuit for AOAPWM is presented and the circuit of figure 6.3.4.b (FIFO interfacing) is reproduced in action in two examples of input circuitry. In the following pages the following circuits can be found (in order) :

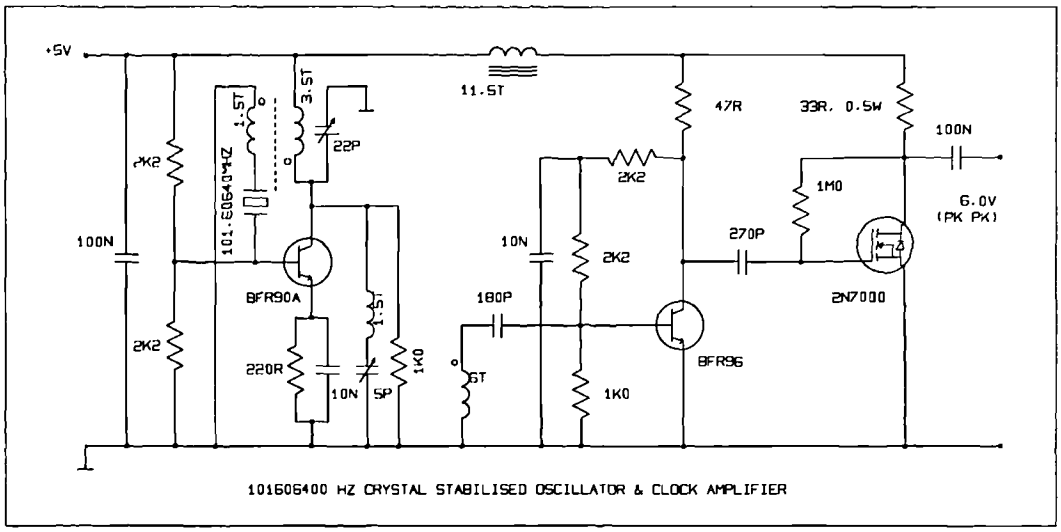
A.7.1 : XTAL & AOAPWM pre-processing schematics

A.7.2 : Re-synchroniser and mono input interface photo

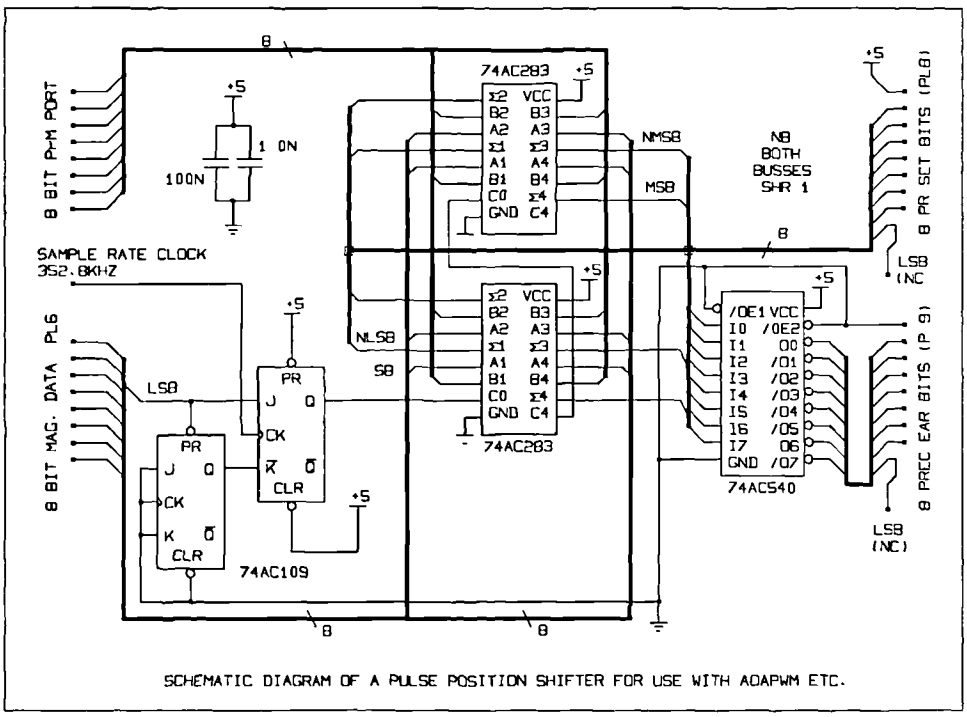
A.7.3 : A stereo input interface schematic & photo

A.7.4 : A cascaded fourth order sinusoidal noise shaper schematic & photo

A.7.5 : Two H-bridge output switch schematics

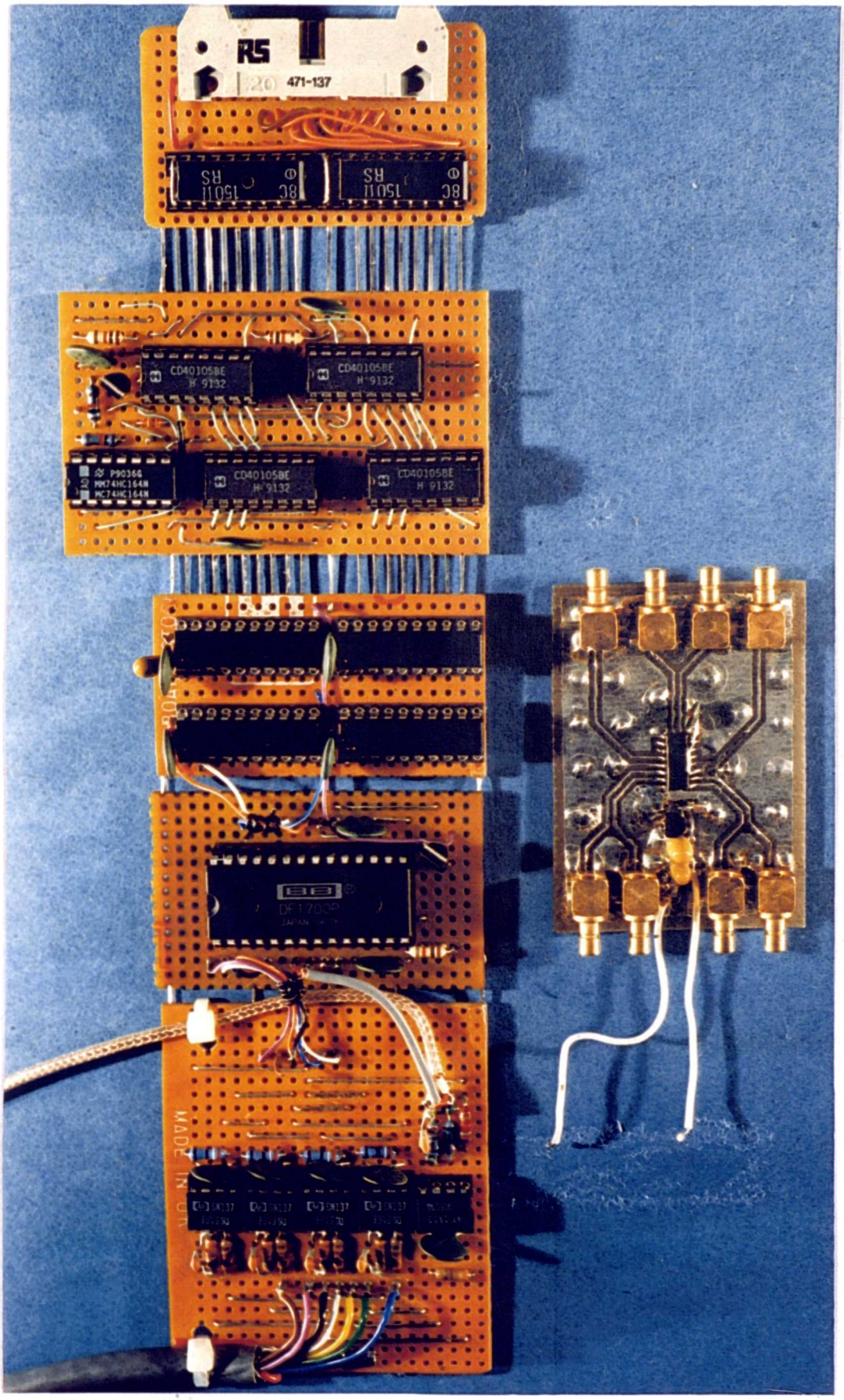


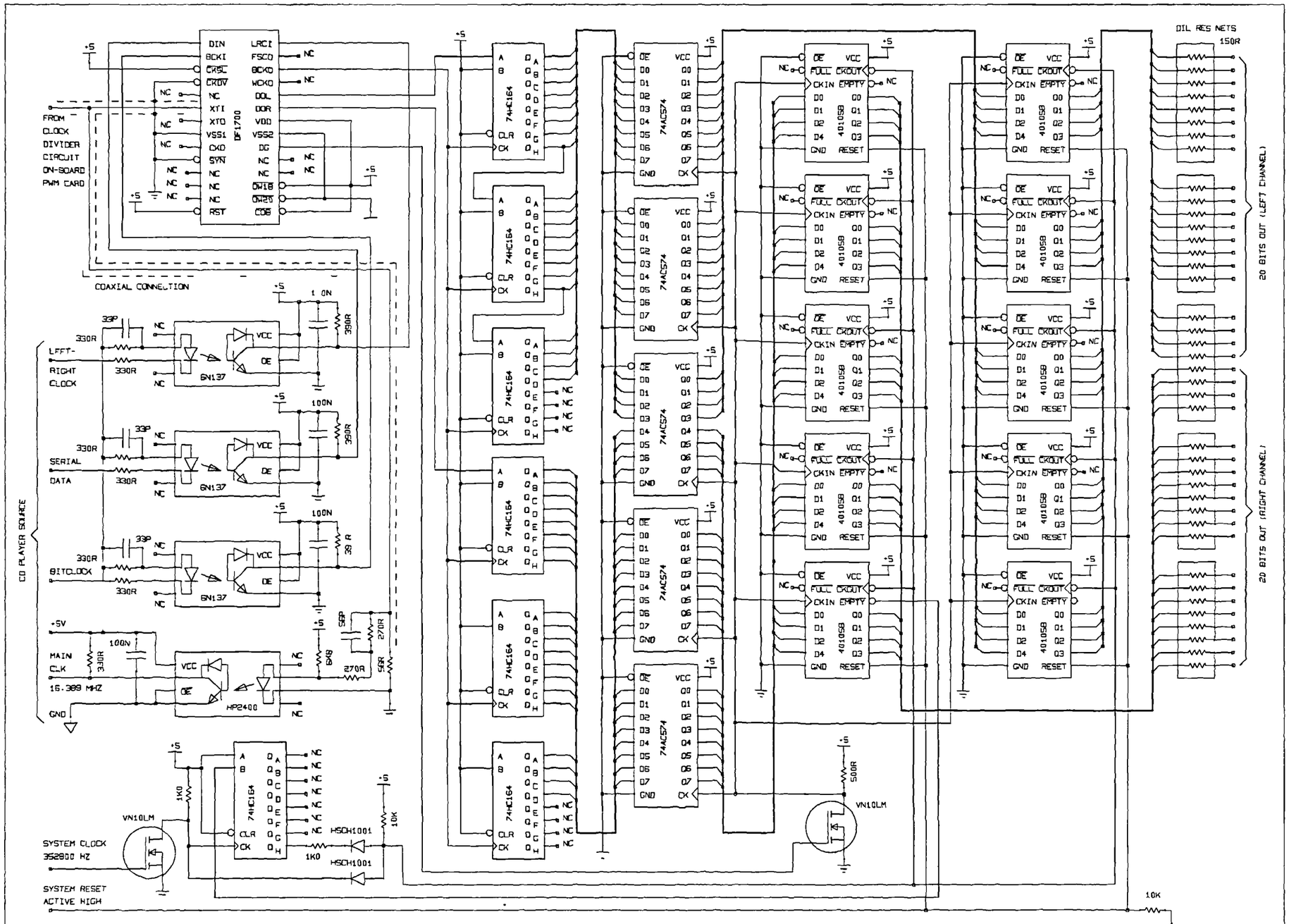
101606400 HZ CRYSTAL STABILISED OSCILLATOR & CLOCK AMPLIFIER



SCHEMATIC DIAGRAM OF A PULSE POSITION SHIFTER FOR USE WITH AOAPWM ETC.

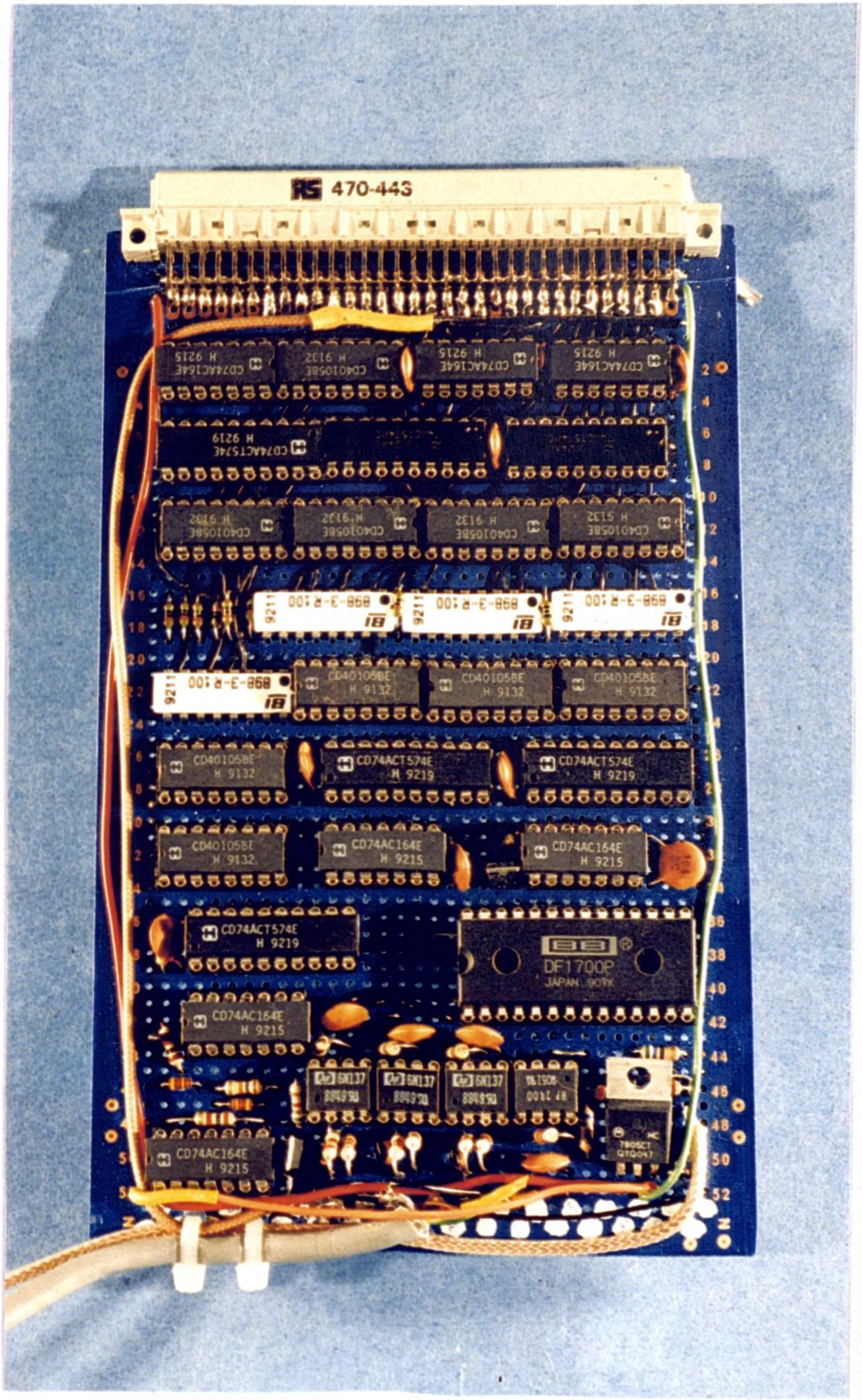
Mono Input Interface & Output Re-synchronisers

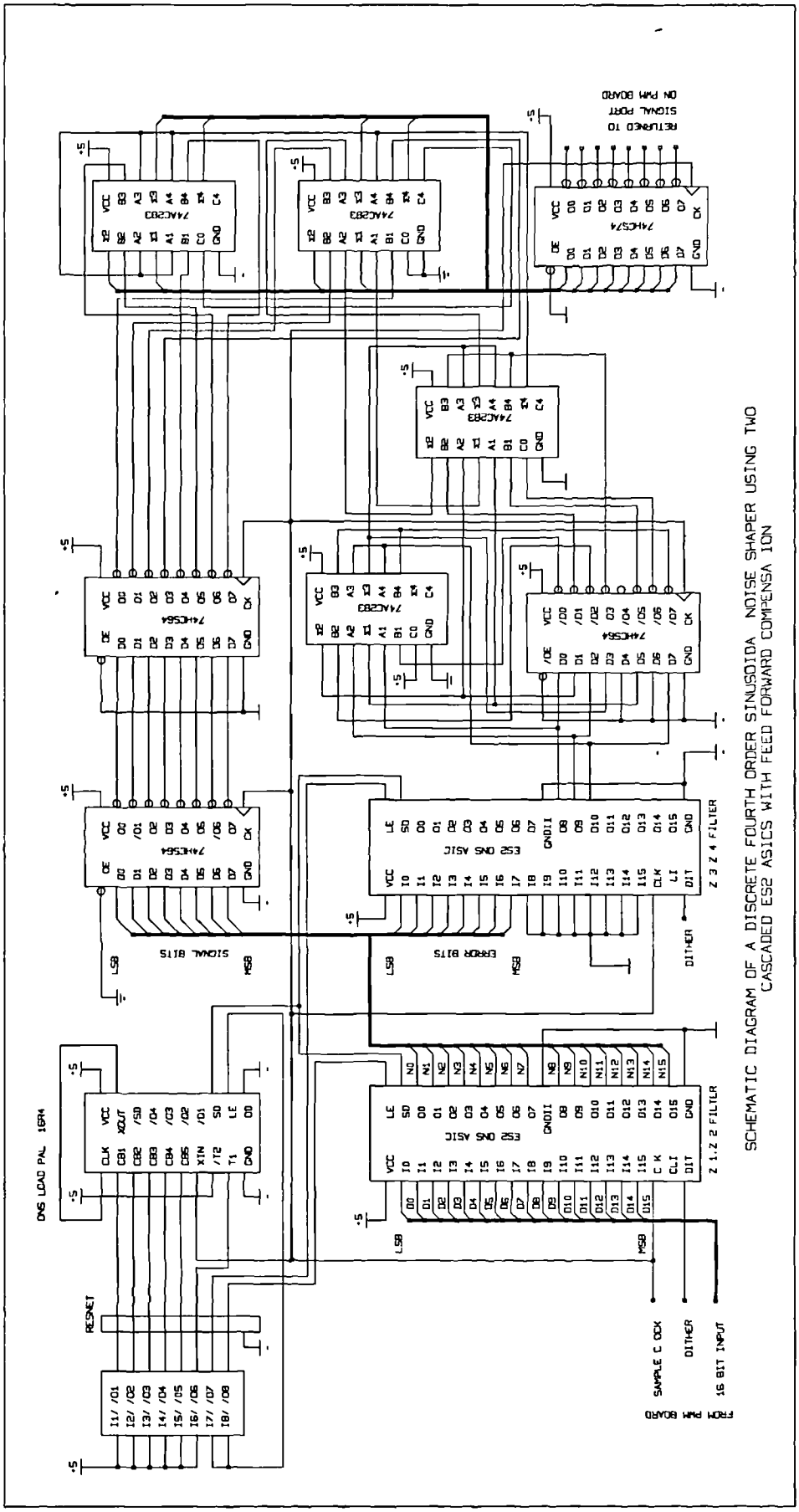




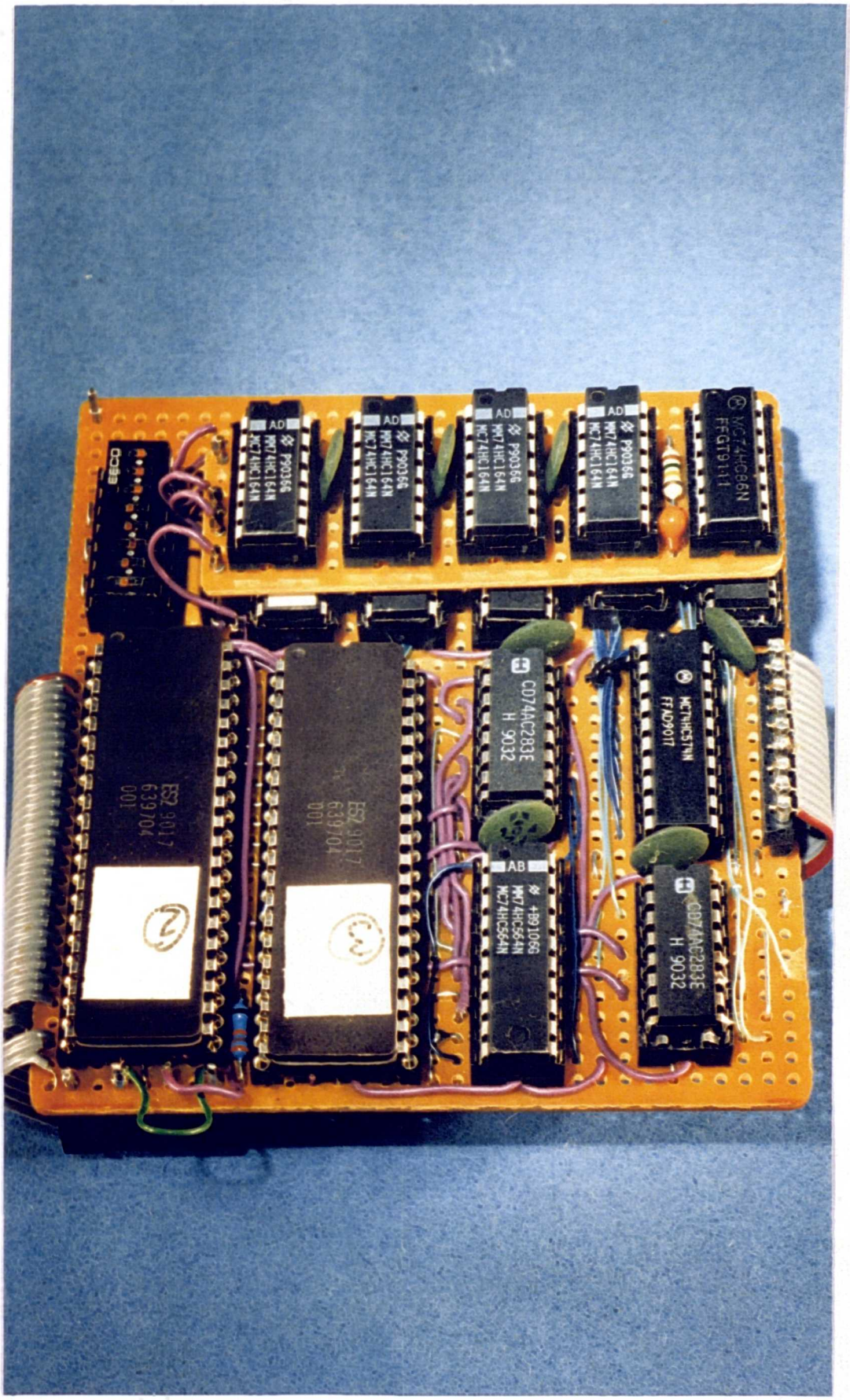
INPUT INTERFACE CIRCUITRY INCLUDING OPTICAL ISOLATION, 8X INTERPOLATION,
SERIES TO PARALLEL CONVERSION, FIFO BUFFERING, AND CURRENT LIMITED LINE DRIVING.

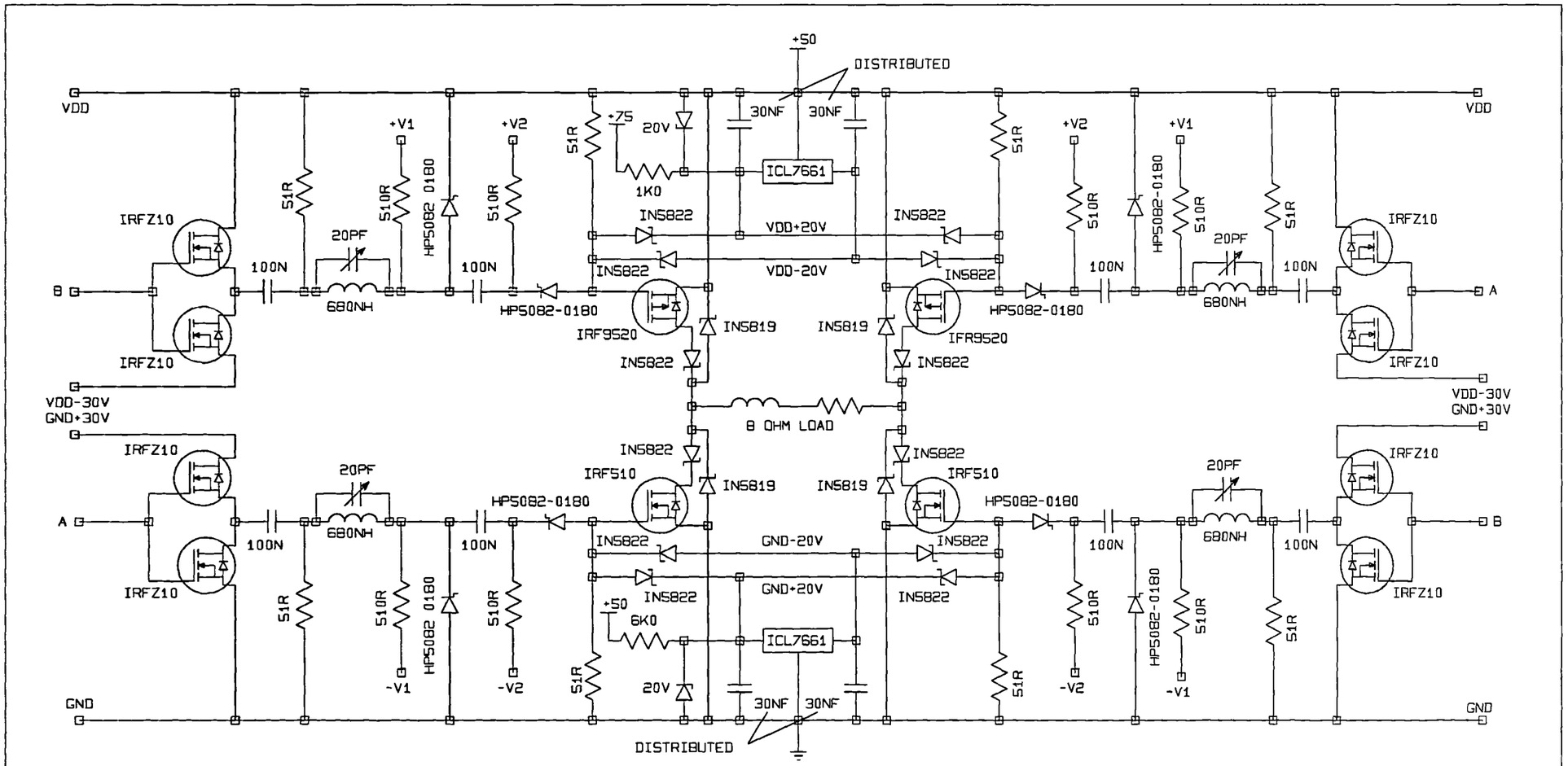
Stereo Input Interface (opto-isolation, S/P conversion, 8x interpolation, 16 stage FIFO)



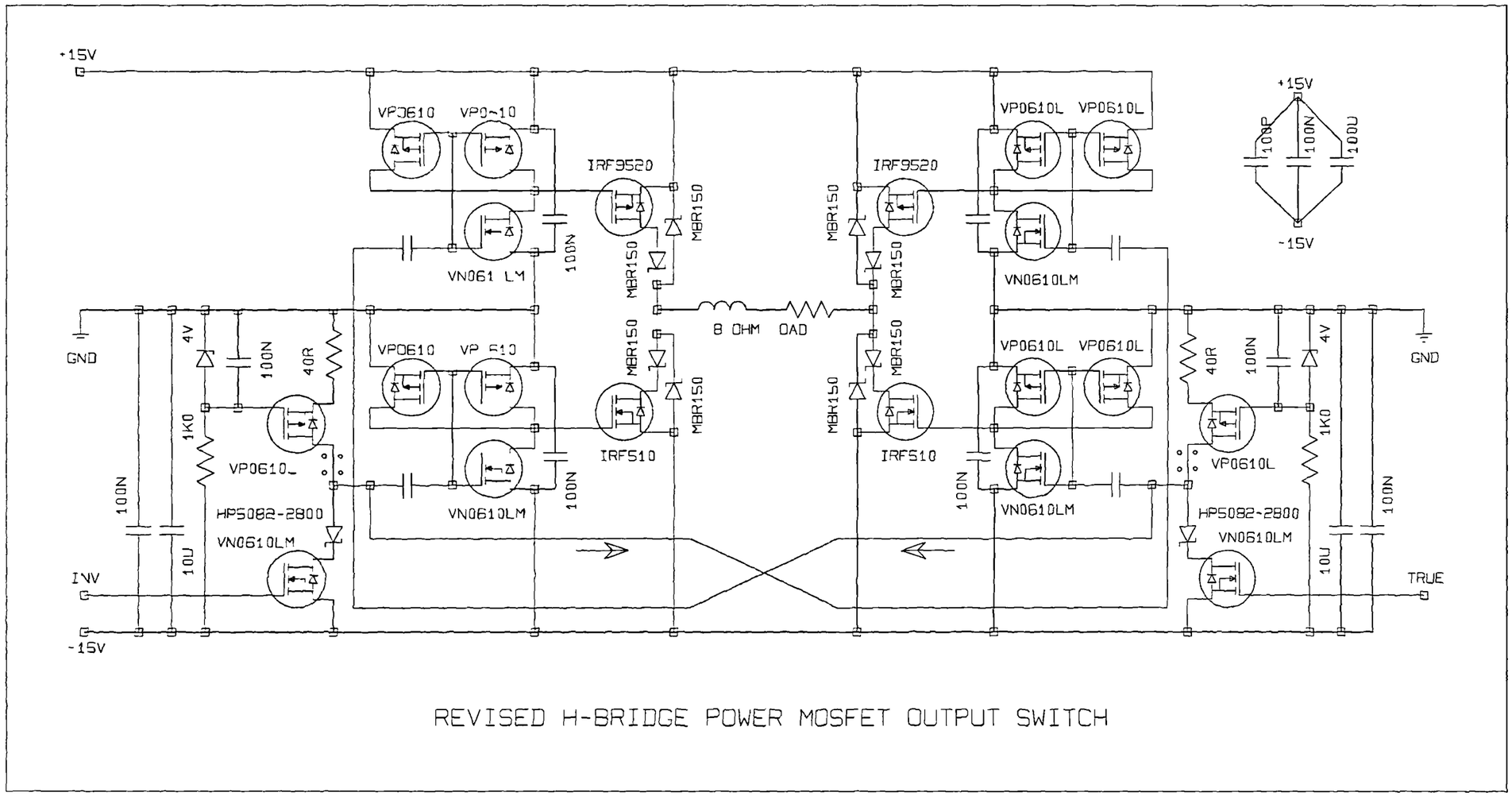

SCHEMATIC DIAGRAM OF A DISCRETE FOURTH ORDER SINUSOIDA NOISE SHAPER USING TWO CASCADED ES2 ASICS WITH FEED FORWARD COMPENSA ION
Z 1.2 2 FILTER
Z 3 Z 4 FILTER
ES2 ONS ASIC
74AC283
74HCS64
74HCS74
SIGNAL BITS
ERROR BITS
DITHER
SAMPLE C OCK
16 BIT INPUT
FROM PWM BOARD
RETURNED TO
SIGNAL PORT
ON PWM BOARD
RESNET
ONS LOAD PAL 16R4
LSB
MSB
+5
-5

Cascaded Fourth Order Noise Shaper (with PRBS dither)





INITIAL POWER MOSFET SWITCH BASED ON AN H-BRIDGE CONFIGURATION IN THE OUTPUT



REVISED H-BRIDGE POWER MOSFET OUTPUT SWITCH

Appendix 8 : The Derivation & Evaluation of PWM Spectra.

A.8.1 : Overview of the analysis

In this appendix the algebraic analysis and preparation for numeric evaluation of theoretical spectra from various PWM techniques will be presented. All this work concentrates on the continuous-time description of unquantised PWM responses to tonal stimuli. When applied to DPWMs this approach is expected to be valid provided the quantisation noise is small and a higher sampling rate than 2.pi times the highest signal frequency is used (see equation 2.2.1.b).

Initially, a mathematical model developed by Black [BLA53] and suitable for this analysis will be described, before the technique of Double Fourier Analysis, as suggested by Bennett [BLA53], is applied to naturally sampled, trailing edged modulation (natural TEPWM). From this minor changes of variables will be used to arrive at theoretical spectra for naturally sampled, leading edged modulation (natural LEPWM) and double sided modulation (DSPWM). Later, with modification to the original model and a change of integration field, the spectrum for uniformly sampled variants of these modulation types (uniform TEPWM, uniform LEPWM & SYMPWM) will be found in a similar way. With one final variable substitution, the spectrum of two sample consecutive pulse width modulation (2SCPWM) will be presented. In each of the uniformly sampled modulation types, the output will still be assumed to be represented in continuous time despite the sampled nature of the input signal but amplitude quantisation will be ignored.

Having completed the algebraic analysis of PWM types, the spectrum found by using a first order approximation to natural sampling (weighted average pre-compensation : WAPWM) will be prepared. This solution cannot be reduced to Bessel functions as the previous types can, so after simplification by algebraic means, numeric integration must be used to provide a solution. With this the spectrum of PWM using enhanced sampling (ESPWM), [MEL91] can be derived. It is interesting to note that this leads to lower harmonic distortion results than previously published [CHE92] as a result of properly accounting for the poor approximation of natural sampling rather than using the ideal spectrum of natural sampling itself [PAU92A].

Similar mixtures of analysis and evaluation can be used for assessing the performance of other modulation schemes such as PNPWM [GOL92] or LPWM [PED94], but the complexity of this technique limits its usefulness. Computer simulation is proving increasingly accurate and fast, providing not only consistent results with the theory presented in this section, but also further details about the nature of response to quantisation noise, noise shaped noise and multiple input tones. For these reasons, further work in this area is pointed towards simulation, and this analysis is included here as an appendix to provide a basis for the conclusions drawn in chapter 2, and thereafter.

A.8.2 : Derivation of Natural TEPWM's spectrum using the wavy wall model

In order to derive the spectrum for any modulation a suitable model of the system is required. Pulse width modulation can be derived using a double Fourier series and the three dimension model used below is suitable for this approach. From this model the necessary equations will be constructed for analysing trailing edge naturally sampled pulse width modulation.

Consider parallel walls, each of unit height, with flat left sides but with thickness, 'x', varying as a sinusoid along their length, y:-

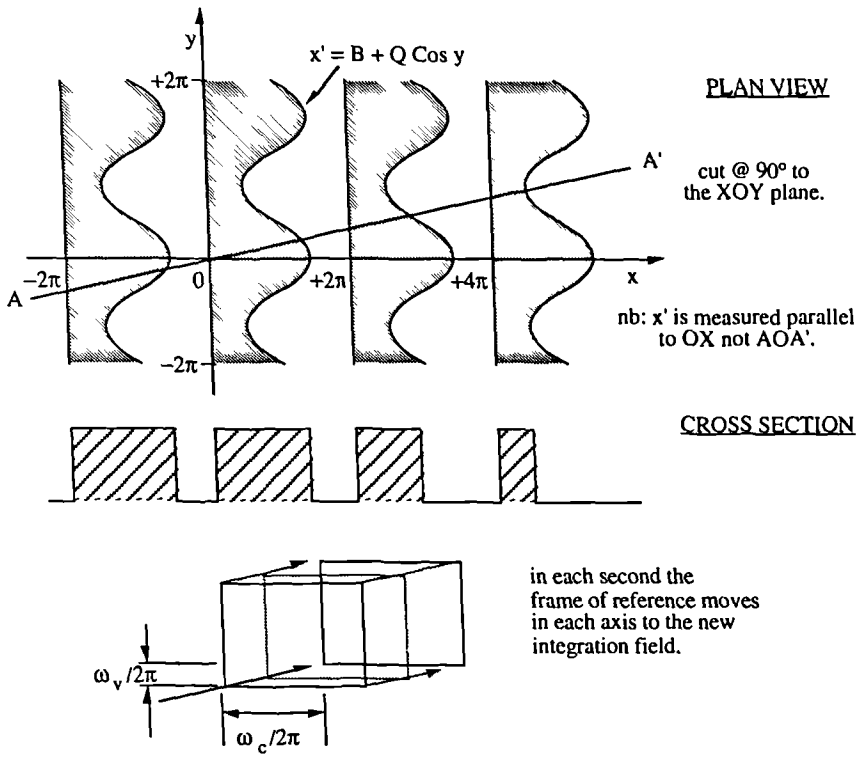


Figure A.8.2.a : Views of a Series of Varying Thickness Walls : 'The Wavy Wall Model'

For simplicity later we should redefine the ratio of the unmodulated pulse width over the repetition period to be 'k'; also for a maximum range of modulation of +/- π, and a deviation of +/- 'Q', we can define $Q = M\pi$, where M% is the modulation index. If we consider x to be repeating as $\omega_c t$, and y to be repeating as $\omega_v t$, we can describe the thickness at any point as: $x' = B + Q\ Cos\ y$, (nb.: parallel to OX not AOA') where: $B = 2\pi k$, and $Q = M\pi$.

For each cross section parallel to OX, the pulses all have the same duration and are periodic, so for any specific value 'y_1' we can re-express this as a Fourier series:-

$$F(x,y_1) = a_0 y_1 + \sum_{m=1}^{\infty} \{ a_m(y_1) Cos(mx) + b_m(y_1) Sin(mx) \}$$

Eqn A.8.2.a

where :

$$a_m(y_1) = \frac{1}{\pi} \int_0^{2\pi} F(x,y_1)\, Cos(mx)\, dx \qquad (m = 0,1,2....)$$

Eqn A.8.2.b

and similarly,

$$b_{m(y_1)} = \frac{1}{\pi}\int_0^{2\pi} F(x,y_1)\,Sin(mx)\,dx \qquad (m=1,2,3....)$$

Eqn A.8.2.c

$a_m(y_1)$ and $b_m(y_1)$ depend on the value of y_1, ie.: these are functions of 'y'. Values of 'y' are periodic for tonal input, so these in turn can each be represented as a Fourier series. For example :

$$a_{mn(y)} = \frac{1}{2}c_{0m} + \sum_{n=1}^{\infty}\{c_{nm}\,Cos(ny) + d_{nm}\,Sin(ny)\}$$

Eqn A.8.2.d

where :

$$c_{nm} = \frac{1}{\pi}\int_0^{2\pi} a_m(y)\,Cos(ny)\,dy \quad , \quad (n=0,1,2...)$$

Eqn A.8.2.e

and,

$$d_{nm} = \frac{1}{\pi}\int_0^{2\pi} a_m(y)\,Sin(ny)\,dy \quad , \quad (n=1,2,3...)$$

Eqn A.8.2.f

similarly :

$$b_{mn(y)} = \frac{1}{2}e_{0m} + \sum_{n=1}^{\infty}\{e_{nm}\,Cos(ny) + g_{nm}\,Sin(ny)\}$$

Eqn A.8.2.g

where :

$$e_{nm} = \frac{1}{\pi}\int_0^{2\pi} b_m(y)\,Cos(ny)\,dy \quad , \quad (n=0,1,2...)$$

Eqn A.8.2.h

and,

$$g_{nm} = \frac{1}{\pi}\int_0^{2\pi} b_m(y)\,Sin(ny)\,dy \quad , \quad (n=1,2,3...)$$

Eqn A.8.2.i

Using these equations, a_m and b_m can be eliminated and F(x,y) can be expressed in terms of the volume integral across x and y of standard trigonometric functions. The double integral can be evaluated for the simple cases (where m or n is zero) by direct integration since F(y) = 1, $0 < x < B+Q\,Cos(y)$; for other values it can be re-expressed in complex exponentials and reorganised as the standard form of a Bessel function of the first kind.

To achieve this, a_m and b_m must be eliminated from F(x,y) by substituting equations A.8.2.d and A.8.2.g into equation A.8.2.a :

$$F(x,y) = \frac{1}{2}a_0(y) + \sum_{m=1}^{\infty}\{[\frac{1}{2}c_{0m} + \sum_{n=1}^{\infty}(c_{nm}\,Cos(ny) + d_{nm}\,Sin(ny))]\,Cos(mx) + [\frac{1}{2}e_{0m} + \sum_{n=1}^{\infty}(e_{nm}\,Cos(ny) + g_{nm}\,Sin(ny))]\,Sin(mx)\}$$

Multiplying out the trigonometric terms yields :

$$F(x,y) = \frac{1}{2}a_0(y) + \sum_{m=1}^{\infty}\{\frac{1}{2}c_{0m}Cos(mx) + \frac{1}{2}e_{0m}Sin(mx) + \frac{1}{2}.\sum_{n=1}^{\infty}[c_{nm}Cos(ny).Cos(mx) + d_{nm}Sin(ny).Cos(mx)] + \frac{1}{2}.\sum_{n=1}^{\infty}[e_{nm}Cos(ny).Sin(mx) + g_{nm}Sin(ny).Sin(mx)]\}$$

from which, expanding the trigonometric products,

$$F(x,y) = \frac{1}{2}a_0(y) + \sum_{m=1}^{\infty}\{\frac{1}{2}c_{0m}Cos(mx) + \frac{1}{2}e_{0m}Sin(mx) + \frac{1}{2}.\sum_{n=1}^{\infty}[\ c_{nm}(Cos(ny+mx)+Cos(ny-mx)) + d_{nm}(Sin(ny+mx)+Sin(ny-mx)) + e_{nm}(Sin(ny+mx)-Sin(ny-mx)) + g_{nm}(Cos(ny-mx)-Cos(ny+mx))]\}$$

and collecting terms, yields :

$$F(x,y) = \frac{1}{2}a_0(y) + \sum_{m=1}^{\infty}\{\frac{1}{2}c_{0m}Cos(mx) + \frac{1}{2}e_{0m}Sin(mx) + \frac{1}{2}.\sum_{n=1}^{\infty}[\ (c_{nm}-g_{nm}).Cos(mx+ny) + (c_{nm}+g_{nm}).Cos(mx-ny) + (e_{nm}+d_{nm}).Sin(mx+ny) + (e_{nm}-d_{nm}).Sin(mx-ny)]\}$$

Eqn A.8.2.j

Having eliminated a_m and b_m, the sub-series coefficients (c_{nm}, d_{nm}, e_{nm} and g_{nm}) need to be found. By substituting A.8.2.b into A.8.2.e, c_{nm} can be expressed as the double integral :

$$c_{nm} = \frac{1}{\pi}\int_0^{2\pi}\{\frac{1}{\pi}\int_0^{2\pi}F(x,y)\,Cos(mx)\,dx\}\,Cos(ny)\,dy$$

$$= \frac{1}{\pi^2}\int_0^{2\pi}\int_0^{2\pi}F(x,y)\,Cos(mx).Cos(ny)\,dx\,dy$$

$$= \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi}F(x,y)\,[Cos(mx+ny)+Cos(mx-ny)]\,dx\,dy$$

hence equation A.8.2.k :

$$c_{nm} = \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi}F(x,y)\,Cos(mx+ny)\,dx\,dy + \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi}F(x,y)\,Cos(mx-ny)\,dx\,dy$$

and similarly equations A.8.2.l, A.8.2.m and A.8.2.n :

$$d_{nm} = \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Sin(mx+ny)\,dx\,dy - \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Sin(mx-ny)\,dx\,dy$$

Eqn A.8.2.l

$$e_{nm} = \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Sin(mx+ny)\,dx\,dy + \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Sin(mx-ny)\,dx\,dy$$

Eqn A.8.2.m

$$g_{nm} = \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Cos(mx-ny)\,dx\,dy - \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Cos(mx+ny)\,dx\,dy$$

Eqn A.8.2.n

From equations A.8.2.k, A.8.2.l, A.8.2.m and A.8.2.n, the terms for the second summation in equation A.8.2.j can be calculated, thus :

$$c_{nm} - g_{nm} = \frac{1}{\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Cos(mx+ny)\,dx\,dy$$

$$c_{nm} + g_{nm} = \frac{1}{\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Cos(mx-ny)\,dx\,dy$$

$$e_{nm} + d_{nm} = \frac{1}{\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Sin(mx+ny)\,dx\,dy$$

$$e_{nm} - d_{nm} = \frac{1}{\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Sin(mx-ny)\,dx\,dy$$

Also, from equation A.8.2.k with n=0, the coefficients for the first summation in equation A.8.2.j can be found :

$$\frac{1}{2}c_{0m} = \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Cos(mx)\,dx\,dy$$

and from equation A.8.2.m, with n=0 :

$$\frac{1}{2}e_{0m} = \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Sin(mx)\,dx\,dy$$

Lastly, from equation A.8.2.d with m=0, the first term in equation A.8.2.j can be found :

$$a_0(y) = \frac{1}{2}c_{00} + \sum_{n=1}^{\infty}\{c_{n0}\,Cos(ny) + d_{n0}\,Sin(ny)$$

which when expanded, in the double integral form, can be simplified to :

$$a_0(y) = \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,dx\,dy + \sum_{n=1}^{\infty}\{\frac{1}{\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Cos(ny)\,dx\,dy\,Cos(ny) + \frac{1}{\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Sin(ny)\,dx\,dy\,Sin(ny)\}$$

Using all of these substituted back into equation A.8.2.j, yields the generalised trigonometric representation of the output spectrum for naturally sampled pulse width modulation with any periodic input signal :

$$F(x,y) = \frac{1}{2}[\frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,dx\,dy]$$

$$+\sum_{n=1}^{\infty}\frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)Cos(ny)dx\,dy.Cos(ny) + \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)Sin(ny)dx\,dy.Sin(ny)$$

$$+\sum_{m=1}^{\infty}\frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)Cos(mx)dx\,dy.Cos(mx) + \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)Sin(mx)dx\,dy.Sin(mx)$$

$$+\sum_{m=1}^{\infty}\sum_{n=1}^{\infty}[\frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Cos(mx+ny)\,dx\,dy\,.\,Cos(mx+ny)$$
$$+\frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Cos(mx-ny)\,dx\,dy\,.\,Cos(mx-ny)$$
$$+\frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Sin(mx-ny)\,dx\,dy\,.\,Sin(mx+ny)$$
$$+\frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Sin(mx-ny)\,dx\,dy\,.\,Sin(mx-ny)]$$

Defining :

$$A_{mn} = \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Cos(mx+ny)\,dx\,dy$$

and :

$$B_{mn} = \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Sin(mx+ny)\,dx\,dy$$

the spectrum can be simplified as :

$$F(x,y) = \frac{1}{2}A_{00} + \sum_{n=1}^{\infty}\{ A_{0n}Cos(ny) + B_{0n}Sin(ny)\} + \sum_{m=1}^{\infty}\{ A_{m0}Cos(mx) + B_{m0}Sin(mx)\} + \sum_{m=1}^{\infty}\sum_{n=\pm 1}^{\pm\infty}\{ A_{mn}Cos(mx+ny) + B_{mn}Sin(mx+ny)\}$$

Eqn A.8.2.o

The four lines of this representation of the modulated spectrum represent (from top to bottom) the DC content of the signal, the modulated signal and its harmonics, the modulating signal and its harmonics (the 'carrier' or PRR) and the signal sidebands about each carrier harmonic.

Evaluating A_{mn} and B_{mn} can be made easier if they are expressed in complex exponential form instead of the trigonometric form, using the identity :

$$e^{i\theta} = Cos(\theta) + i.Sin(\theta)$$

so that :

$$A_{mn} + iB_{mn} = \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{2\pi} F(x,y)e^{i(mx+ny)}dx\,dy$$

Taking the 'walls' in this model to be of unity height between x = 0 and x = B+Q.*Cos*(y), ie. trailing edged modulation (the height can be thought of as a scaling factor representing the output voltage rail separation) and taking the 'ground' to be zero height outside this range, ie. two level or 'AD' modulation [MAR70] gives :

$$A_{mn} + iB_{mn} = \frac{1}{2\pi^2}\int_0^{2\pi}\int_0^{B+QCos(y)} F(x,y)e^{i(mx+ny)}dx\,dy$$

$$= \frac{1}{2\pi^2 im}\int_0^{2\pi}\{ e^{i(mx+ny)}\Big|_0^{B+QCos(y)}\}dy$$

$$= -\frac{i}{2\pi^2 m}\int_0^{2\pi}\{ e^{(imB+imQCos(y)+iny)} - e^{(iny)}\}dy$$

Eqn A.8.2.p

nb. : $e^{iny} = Cos(ny) + iSin(ny) \Rightarrow \int_0^{2\pi} e^{iny}\,dy = 0$, for any value of $n \neq 0$

Since 'm' is in the denominator of equation A.8.2.p above, this expression becomes indeterminate for cases where m = 0; however, all the summations in equation A.8.2.o avoid the m = 0 case, so this equation can be re-expressed using :

$$A_{mn} + i B_{mn} = -\frac{i}{2\pi^2 m} e^{imB} \int_0^{2\pi} e^{imQCos(y)} . e^{iny}\, dy$$

This is equivalent to a Bessel function of the first kind, and integer order, related by the identities :

$$J_n(z) = \frac{i^{-n}}{2\pi} \int_0^{2\pi} e^{izCos(\Phi)} . e^{in\Phi}\, d\Phi \quad , \text{ and } \quad e^{i\frac{n\pi}{2}} = i^n$$

hence :

$$A_{mn} + i B_{mn} = -\frac{i}{2\pi^2 m} e^{imB} . 2\pi i^n J_n(mQ) = -\frac{i}{2\pi^2 m} e^{i(mB + \frac{n\pi}{2})} J_n(mQ)$$

and equating real and imaginary parts :

$$A_{mn} = \frac{1}{\pi m} J_n(mQ) . Sin(mB + \frac{n\pi}{2})$$

$$B_{mn} = \frac{-1}{\pi m} J_n(mQ) . Cos(mB + \frac{n\pi}{2})$$

Eqn A.8.2.q

For the special cases of 'm' or 'n' equal to zero, the integrals for A_{mn} and B_{mn} can be evaluated knowing that F(x,y) = 1, and $B+QCos(y) > x > 0$. For equation A.8.2.o, the Bessel expression for the double summation and five special valued coefficients are required : A_{0n} and B_{0n} , for 'n' a positive integer, A_{m0} and B_{m0} , for 'm' a positive integer and A_{00} :

$$A_{00} = \frac{1}{2\pi^2} \int_0^{2\pi} \int_0^{B+QCos(y)} F(x,y)\, dx\, dy = \frac{1}{2\pi^2} \int_0^{2\pi} B + Q\,Cos(y)\, dy$$

$$= \frac{1}{2\pi^2} [\, By + Q.Sin(y)\,]_0^{2\pi} = \frac{B}{k} = 2k$$

Eqn A.8.2.r

Next, evaluating A_{0n} :

$$A_{0n} = \frac{1}{2\pi^2} \int_0^{2\pi} \int_0^{B+QCos(y)} F(x,y)\, Cos(ny)\, dx\, dy$$

$$= \frac{1}{2\pi^2} \int_0^{2\pi} BCos(ny) + QCos(y).Cos(ny)\, dy$$

$$= \frac{1}{2\pi^2} \int_0^{2\pi} BCos(ny) + \frac{Q}{2} Cos(n+1)y + \frac{Q}{2} Cos(n-1)y\, dy$$

$$= \frac{1}{2\pi^2} [\frac{B}{n} Sin(ny) + \frac{Q}{2(n+1)} Sin(n+1)y + \frac{Q}{2(n-1)} Sin(n-1)y\,]_0^{2\pi}$$

Integrals of the sine function over complete cycles will be zero for all integer values of 'n', but the last term becomes indeterminate as a result of the 'n-1' term in its coefficient denominator, so for A_{0n}, only A_{01} need be evaluated :

$$A_{01} = \frac{1}{2\pi^2} \int_0^{2\pi} \int_0^{B+QCos(y)} F(x,y)\, Cos(y)\, dx\, dy$$

$$= \frac{1}{2\pi^2} \int_0^{2\pi} B\,Cos(y) + Q\,Cos^2(y)\, dy$$

$$= \frac{1}{2\pi^2} \int_0^{2\pi} B\,Cos(y) + Q.\frac{1}{2}[1 + Cos(2y)]\, dy$$

$$= \frac{1}{2\pi^2} \left[B\,Sin(y) + \frac{Q}{2}y + \frac{Q}{4}Sin(2y) \right]_0^{2\pi}$$

$$= \frac{1}{2\pi^2} . Q . \pi = \frac{Q}{2\pi} = \frac{M}{2}$$

Eqn A.8.2.s

Next, evaluating B_{0n} :

$$B_{0n} = \frac{1}{2\pi^2} \int_0^{2\pi} \int_0^{B+QCos(y)} F(x,y)\, Sin(ny)\, dx\, dy$$

$$= \frac{1}{2\pi^2} \int_0^{2\pi} B\,Sin(ny) + Q\,Cos(y).Sin(ny)\, dy$$

$$= \frac{1}{2\pi^2} \int_0^{2\pi} B\,Sin(ny) + \frac{Q}{2}Sin(n+1)y + \frac{Q}{2}Sin(n-1)y\, dy$$

$$= \frac{1}{2\pi^2} \left[-\frac{B}{n}Cos(ny) - \frac{Q}{2(n+1)}Cos(n+1)y - \frac{Q}{2(n-1)}Cos(n-1)y \right]_0^{2\pi}$$

As in the evaluation of A_{0n}, the integrals of the trigonometric functions over complete cycles become zero, so only when a coefficient becomes indeterminate does it need to be evaluated, specifically, B_{01} :

$$B_{01} = \frac{1}{2\pi^2} \int_0^{2\pi} \int_0^{B+QCos(y)} F(x,y)\, Sin(y)\, dx\, dy$$

$$= \frac{1}{2\pi^2} \int_0^{2\pi} B\,Sin(y) + Q\,Cos(y).\,Sin(y)\,dy$$

$$= \frac{1}{2\pi^2} \int_0^{2\pi} B\,Sin(y) + Q.\frac{1}{2}\,Sin(2y)\,dy$$

$$= \frac{1}{2\pi^2} \left[-B\,Cos(y) - \frac{Q}{4}\,Cos(2y) \right]_0^{2\pi} = 0$$

Eqn A.8.2.t

Next, evaluating the carrier and its harmonics :

$$A_{m0} + iB_{m0} = \frac{1}{2\pi^2} \int_0^{2\pi}\int_0^{2\pi} F(x,y)\,Cos(mx) + i\,Sin(mx)\,dx\,dy$$

$$= \frac{1}{2\pi^2} \int_0^{2\pi}\int_0^{B+Q\,Cos(y)} e^{imx}\,dx\,dy$$

$$= \frac{1}{2\pi^2 im} \int_0^{2\pi} e^{im(B+Q\,Cos(y))} - e^0\,dy$$

$$= \frac{1}{2\pi^2 im} \int_0^{2\pi} e^{imB + imQ\,Cos(y)}\,dy - 2\pi$$

$$= \frac{-i}{m\pi} e^{imB} . \frac{1}{2\pi} \int_0^{2\pi} e^{imQ\,Cos(y)} . e^0\,dy + \frac{i}{m\pi}$$

$$= \frac{-i}{m\pi} e^{imB} . J_0(mQ) + \frac{i}{m\pi}$$

$$= \frac{-i}{m\pi} \left[J_0(mQ).(Cos(mB) + i\,Sin(mB)) - 1 \right]$$

$$= \frac{1}{m\pi} \left[J_0(mQ).Sin(mB) - i.(J_0(mQ).Cos(mB) - 1) \right]$$

Separating the real and imaginary parts, yields the values of coefficients A_{m0} and B_{m0} :

$$A_{m0} = \frac{1}{m\pi} J_0(mQ).Sin(mB)$$

$$B_{m0} = \frac{1}{m\pi} \left[1 - J_0(mQ).Cos(mB) \right]$$

Substituting these values into the third line of equation A.8.2.o (the second summation) yields :

$$A_{m0}\,Cos(mx)+B_{m0}\,Sin(mx)=\frac{J_0(mQ)}{m\pi}[Sin(mB).Cos(mx)-Cos(mB).Sin(mx)]$$

$$+\frac{1}{m\pi}Sin(mx)$$

Expanding the trigonometric products as 'sum and difference' angles allows simpler expression of the variable associated with the Bessel coefficient, since :

$$\begin{aligned}Sin(mB).Cos(mx)-Cos(mB).Sin(mx) &= \frac{1}{2}[Sin(mB+mx)-Sin(mB-mx)\\ &\quad -Sin(mB+mx)-Sin(mB-mx)]\\ &= -Sin(mx-mB)\\ &= -Sin(m\omega_c t-2m\pi k)\end{aligned}$$

thus :

$$A_{m0}\,Cos(mx)+B_{m0}\,Sin(mx)=\frac{1}{m\pi}[Sin(mx)-J_0(mM\pi).Sin(m\omega_c t-2\pi mk)]$$

Eqn A.8.2.u

Finally, simplifying the double summation of equation A.8.2.o :

$$\sum_{m=1}^{\infty}\sum_{n=\pm1}^{\pm\infty}\frac{J_n(mQ)}{m\pi}[Sin(mB+\frac{n\pi}{2}).Cos(mx+ny)-Cos(mB+\frac{n\pi}{2}).Sin(mx+ny)]$$

$$=\sum_{m=1}^{\infty}\sum_{n=\pm1}^{\pm\infty}\frac{J_n(mQ)}{m\pi}.\frac{1}{2}[Sin(mB+\frac{n\pi}{2}+mx+ny)-Sin(mx+ny-mB-\frac{n\pi}{2})$$
$$-Sin(mB+\frac{n\pi}{2}+mx+ny)-Sin(mx+ny-mB-\frac{n\pi}{2})]$$

$$=\sum_{m=1}^{\infty}\sum_{n=\pm1}^{\pm\infty}\frac{J_n(mQ)}{m\pi}.[-Sin(mx+ny-mB-\frac{n\pi}{2})]$$

$$=-\sum_{m=1}^{\infty}\sum_{n=\pm1}^{\pm\infty}\frac{J_n(mQ)}{m\pi}Sin(m\omega_c t+n\omega_v t-2m\pi k-\frac{n\pi}{2})$$

Eqn A.8.2.v

Substituting, equations A.8.2.s, A.8.2.t, A.8.2.u and A.8.2.v into equation A.8.2.o yields the form of the output spectrum as used in chapter 2 (cf. equation 2.3.2.a) :

$$\begin{aligned}F_{NTE}(t) &= k+\frac{M}{2}.Cos(\omega_v t)\\ &\quad+\sum_{m=1}^{\infty}\{\frac{1}{m\pi}.[Sin(m\omega_c t)-J_0(mM\pi).Sin(m\omega_c t-2m\pi k)]\}\\ &\quad-\sum_{m=1}^{\infty}\{\sum_{n=\pm1}^{\infty}\{\frac{J_n(mM\pi)}{m\pi}.[Sin(m\omega_c t+n\omega_v t-2m\pi k-\frac{n\pi}{2})]\}\}\end{aligned}$$

Eqn A.8.2.w

A.8.3 : Derivation of DSPWM's spectrum from Natural TEPWM & LEPWMs'

To derive double sided naturally sampled PWM's spectrum from single sided, a leading edged and trailing edged spectrum can be added. Leading edge modulation's spectrum can be found from trailing edge's by replacing 't' by '-t', yielding :

$$F_{NLE}(t) = k + \frac{M}{2}.Cos(\omega_v t) - \sum_{m=1}^{\infty}\left\{\frac{1}{m\pi}.[Sin(m\omega_c t) - J_0(mM\pi).Sin(m\omega_c t + 2m\pi k)]\right\} + \sum_{m=1}^{\infty}\left\{\sum_{n=\pm 1}^{\infty}\left\{\frac{J_n(mM\pi)}{m\pi}.[Sin(m\omega_c t + n\omega_v t + 2m\pi k + \frac{n\pi}{2})]\right\}\right\}$$

Eqn A.8.3.a

For this addition, the two signals must be appropriately modulated to avoid pulse overlap. The modulation index, 'M', and the modulation offset, 'k' must both be halved. Thus, adding equations A.8.2.w and A.8.3.a :

$$F(x,y) = k + \frac{M}{2}.Cos(\omega_v t) + \sum_{m=1}^{\infty}\left\{\frac{J_0(m\frac{M}{2}\pi)}{m\pi}.[Sin(m\omega_c t + m\pi k) - Sin(m\omega_c t - m\pi k)]\right\} + \sum_{m=1}^{\infty}\sum_{n=\pm 1}^{\pm\infty}\left\{\frac{J_n(m\frac{M}{2}\pi)}{m\pi}.[Sin(m\omega_c t + n\omega_v t + m\pi k + \frac{n\pi}{2}) - Sin(m\omega_c t + n\omega_v t - m\pi k - \frac{n\pi}{2})]\right\}$$

which, after collating constant and variable parts, simplifies to (cf. equation 2.3.5.a) :

$$F_{DSN}(t) = k + \frac{M}{2}Cos(\omega_v t) + \sum_{m=1}^{\infty}\left\{\frac{J_0(mM\frac{\pi}{2})}{m\frac{\pi}{2}}Cos(m\omega_c t)\,Sin(m\pi k)\right\} + \sum_{m=1}^{\infty}\left\{\sum_{n=\pm 1}^{\infty}\left\{\frac{J_n(mM\frac{\pi}{2})}{m\frac{\pi}{2}}Cos(m\omega_c t + n\omega_v t)\,Sin(m\pi k + n\frac{\pi}{2})\right\}\right\}$$

Eqn A.8.3.b

A.8.4 : Derivation of Uniform TEPWM's spectrum

To find uniformly sampled PWM spectra, the same technique of double Fourier analysis can be used with a revised signal : namely the sampled and held version of the continuous-time signal. Since the same comparison signal is used, equation A.8.2.o can be reapplied, with a substitution for the signal variable 'y'. As the signal is constant in the sampling interval (it has been sampled and held), the minimum comparison waveform frequency can be reduced to the Nyquist rate rather than pi times Nyquist as was required in the natural case.

As before, the sampling signal has frequency ω_c and the signal frequency is ω_v so the sampled value, 'y_n', taken from the beginning of the n-th interval occurs at time :

$$t_n = \frac{\omega_c}{\omega_v}.2n\pi \qquad \text{(in } \textit{signal} \text{ -axis units)}$$

the instantaneous signal amplitude at this time would be :

$$v_n = B + Q.Cos(\frac{\omega_v t}{\omega_c t}.2n\pi)$$

This point in the signal waveform occurs before the time of intersection of the sampled signal and the comparison wave by a difference proportional to the sampled signal value, so the following substitutions can be made :

$$y' = y - \frac{\omega_c}{\omega_v}.x \quad , \quad y = \frac{\omega_v}{\omega_c}.x + y' \quad , \quad dy' = dy \qquad \text{Eqn A.8.4.a}$$

hence, revising the limits and the variable of integration from equation A.8.2.p :

$$x = B + Q\,Cos(y')$$

$$A_{mn} + iB_{mn} = \frac{1}{2\pi^2}\int_{-\frac{\omega_v}{\omega_c}x}^{2\pi-\frac{\omega_v}{\omega_c}x}\int_{0}^{B+QCos(y')} e^{i(mx+n[\frac{\omega_v}{\omega_c}x+y'])}\,dx\,dy'$$

and by defining : $G = m + n.\omega_v/\omega_c$ and evaluating the internal integral :

$$= \frac{1}{2\pi^2 iG}\int_{-\frac{\omega_v}{\omega_c}x}^{2\pi-\frac{\omega_v}{\omega_c}x} e^{iG(B+QCos(y')+iny')} - e^{iny'}\,dy' \qquad \text{Eqn A.8.4.b}$$

As in the natural sampling case, integrals of complete periods of the trigonometric functions contribute nothing to the spectrum, and for this sampling type this also permits a useful shift in the limits by $(\omega_v/\omega_c).x$, yielding :

$$A_{mn} + iB_{mn} = -\frac{i}{2\pi^2 G}.e^{iGB}.\int_0^{2\pi} e^{iGQCos(y')}.e^{iny'}\,dy'$$

Expressed in Bessel functions of the first kind, this is equivalent to :

$$A_{mn} + iB_{mn} = -\frac{i}{2\pi^2 G} . e^{iGB} . 2\pi i^n . J_n(GQ)$$

$$= -\frac{i}{G\pi} . e^{i(GB + n\frac{\pi}{2})} . J_n(GQ)$$

Separating real and imaginary parts yields :

$$A_{mn} = \frac{1}{G\pi} . J_n(GQ) . Sin(GB + n\frac{\pi}{2})$$

$$B_{mn} = -\frac{1}{G\pi} . J_n(GQ) . Cos(GB + n\frac{\pi}{2})$$

Eqn A.8.4.c & d

The exceptional terms required for use with equation A.8.2.o can be grouped for evaluation : A_{00}, $A_{0n}+iB_{0n}$, $A_{m0}+iB_{m0}$ and $A_{mn}+iB_{mn}$, and can be evaluated thus :

$$A_{00} = \int_0^{2\pi} \int_0^{B+QCos(y')} F(x,y')\,dx\,dy = \frac{1}{2\pi^2} \int_0^{2\pi} B + QCos(y')\,dy'$$

$$= \frac{1}{2\pi^2} [By' + QSin(y')]_0^{2\pi}$$

$$= \frac{1}{2\pi^2} (2\pi B) = \frac{B}{\pi} = 2k$$

Eqn A.8.4.e

$A_{0n}+iB_{0n}$ can be evaluated directly from equation A.8.4.b after conversion to Bessel form and with the substitution of the value m=0 :

$$A_{0n} + iB_{0n} = -\frac{i}{n\pi \frac{\omega_v}{\omega_c}} . e^{in(\frac{\omega_v}{\omega_c} B + \frac{\pi}{2})} . J_n(nQ\frac{\omega_v}{\omega_c})$$

hence, separating real and imaginary parts :

$$A_{0n} = \frac{\omega_c}{n\pi\omega_v} . J_n(nM\pi\frac{\omega_v}{\omega_c}) . Sin(2n\pi k\frac{\omega_v}{\omega_c} + n\frac{\pi}{2})$$

$$B_{0n} = -\frac{\omega_c}{n\pi\omega_v} . J_n(nM\pi\frac{\omega_v}{\omega_c}) . Cos(2n\pi k\frac{\omega_v}{\omega_c} + n\frac{\pi}{2})$$

so the first summation term from equation A.8.2.o for the uniformly sampled case is :

$$A_{0n} Cos(n\omega_v t) + B_{0n} Sin(n\omega_v t) = \frac{J_n(nM\pi\frac{\omega_v}{\omega_c})}{n\pi\frac{\omega_v}{\omega_c}} . [Sin A . Cos B - Cos A . Sin B]$$

where : $A = 2n\pi k\frac{\omega_v}{\omega_c} + n\frac{\pi}{2}$, and : $B = n\omega_v t$

$$= \frac{J_n(nM\pi\frac{\omega_v}{\omega_c})}{n\pi\frac{\omega_v}{\omega_c}} . [Sin(A-B)] = -\frac{J_n(nM\pi\frac{\omega_v}{\omega_c})}{n\pi\frac{\omega_v}{\omega_c}} . Sin(n\omega_v t - 2n\pi k - n\frac{\pi}{2})$$

Eqn A.8.4.f

To evaluate $A_{m0}+iB_{m0}$, a similar approach to the natural sampling case can be used, knowing :

$$\int_{z}^{2\pi+z} \{ e^{0} \}\, dy = 2\pi$$

which allows equation A.8.4.b to be simplified in the Bessel form for the special case of n=0 :

$$A_{m0} + iB_{m0} = -\frac{i}{m\pi} e^{imB} . J_0(mQ) + \frac{1}{m\pi}$$

$$= \frac{1}{m\pi} . J_0(mQ) . \{ Cos(mB) + i Sin(mB) - 1 \}$$

separating real and imaginary parts yields :

$$A_{m0} = \frac{1}{m\pi} J_0(mQ) Sin(mB)$$

$$B_{m0} = \frac{1}{m\pi} . [1 - J_0(mQ) Cos(mB)]$$

so the second summation term in equation A.8.2.o, for the uniformly sampled case, is :

$$A_{m0} Cos(mx) + B_{m0} Sin(mx) = \frac{J_0(mQ)}{m\pi} . [Sin(mB) . Cos(mx) - Cos(mB) . Sin(mx)]$$

$$+ \frac{1}{m\pi} Sin(mx)$$

$$= \frac{1}{m\pi} . [Sin(m\omega_c t) - J_0(mM\pi) . Sin(m\omega_c t - 2m\pi k)]$$

Eqn A.8.4.g

Lastly, evaluating the double summation term from equation A.8.2.o, using A.8.4.c & d :

$$A_{mn} Cos(mx+ny) + B_{mn} Sin(mx+ny) = \frac{1}{G\pi} J_n(GQ) . [Sin A . Cos B - Cos A . Sin B]$$

where : $A = GB + \frac{n\pi}{2}$, and $B = mx+ny$

$$= - \frac{J_n\left((m+n\frac{\omega_v}{\omega_c})M\pi \right)}{(m+n\frac{\omega_v}{\omega_c})\pi} . Sin [(m\omega_c + n\omega_v)(t - \frac{2\pi k}{\omega_c}) - n\frac{\pi}{2}]$$

Eqn A.8.4.h

and substituting all these terms (A.8.4.e - h) back into equation A.8.2.o, yields the spectrum for uniformly sampled trailing edged modulation (cf. equation 2.3.3.a) :

$$F_{UTE}(t) = k - \sum_{n=1}^{\infty} \left\{ \frac{J_n(\frac{n\pi M\omega_v}{\omega_c})}{\frac{n\pi\omega_v}{\omega_c}} Sin(n\omega_v t - \frac{2n\pi k\omega_v}{\omega_c} - \frac{n\pi}{2}) \right\}$$

$$+ \sum_{m=1}^{\infty} \left\{ \frac{1}{m\pi} . [Sin(m\omega_c t) - J_0(mM\pi) . Sin(m\omega_c t - 2m\pi k)] \right\}$$

$$- \sum_{m=1}^{\infty} \left\{ \sum_{n=\pm 1}^{\pm\infty} \left\{ \frac{J_n[(m\omega_c + n\omega_v)\frac{\pi M}{\omega_c}]}{(m\omega_c + n\omega_v)\frac{\pi}{\omega_c}} Sin[(m\omega_c + n\omega_v)(t - \frac{2\pi k}{\omega_c}) - \frac{n\pi}{2}] \right\} \right\}$$

Eqn A.8.4.i

A.8.5 : Derivation of SYMPWM's spectrum from Uniform TEPWM & LEPWMs'

As in the naturally sampled case, two sided modulation types' spectra can be found from the sum of non-overlapping pulse streams and hence from two single sided modulation types with appropriate modulation indices and offsets. However, in the uniformly sampled case there arises the dilemma of which sampling point should contribute to the current pulse : the signal value at the start of the pulse period, the middle, or the end ?

Two modulation types arise as a result of picking different pairs of trailing and leading edged components. In the simplest case, the signal amplitude at the pulse centre is used for both leading and trailing components yielding a symmetrically placed pulse about the centre of the pulse period : SYMPWM. In the more sophisticated modulation type, the sample from the period start is used for the leading edged component and the sample from the pulse centre is used for the trailing edged component : 2SCPWM. This particular combination permits a higher information rate than SYMPWM (more sampling points), and the cancellation of even order harmonic and sideband terms. In this section the derivation of SYMPWM from uniformly sampled TEPWM and LEPWM will be presented.

From uniformly sampled TEPWM's spectrum, the spectrum for uniformly sampled LEPWM can be found by replacing 't' with '-t'. Thus :

$$F_{ULE}(t) = k + \sum_{n=1}^{\infty}\left\{\frac{J_n\left(\frac{n\pi M\omega_v}{\omega_c}\right)}{\frac{n\pi\omega_v}{\omega_c}} Sin\,(n\omega_v t + \frac{2n\pi k\omega_v}{\omega_c} + \frac{n\pi}{2})\right\}$$

$$- \sum_{m=1}^{\infty}\left\{\frac{1}{m\pi}.[Sin\,(m\omega_c t) - J_0(mM\pi).Sin\,(m\omega_c t + 2m\pi k)]\right\}$$

$$+ \sum_{m=1}^{\infty}\left\{\sum_{n=\pm1}^{\pm\infty}\left\{\frac{J_n[(m\omega_c + n\omega_v)\frac{\pi M}{\omega_c}]}{(m\omega_c + n\omega_v)\frac{\pi}{\omega_c}} Sin\,[(m\omega_c + n\omega_v)(t + \frac{2\pi k}{\omega_c}) + \frac{n\pi}{2}]\right\}\right\}$$

Eqn A.8.5.a

Adding equations A.8.4.i and A.8.5.a, with both modulation index and offset set to one half of their previous value yields :

$$F(t) = k - \sum_{n=1}^{\infty}\left\{\frac{J_n(n\pi\frac{M}{2}\frac{\omega_v}{\omega_c})}{n\pi\frac{\omega_v}{\omega_c}}[Sin\,(n\omega_v t - n\pi k\frac{\omega_v}{\omega_c} - n\frac{\pi}{2}) - Sin\,(n\omega_v t + n\pi k\frac{\omega_v}{\omega_c} + n\frac{\pi}{2})]\right\}$$

$$- \sum_{m=1}^{\infty}\left\{\frac{J_0(m\frac{M}{2}\pi)}{m\pi}[Sin\,(m\omega_c t - m\pi k) - Sin\,(m\omega_c t + m\pi k)]\right\}$$

$$- \sum_{m=1}^{\infty}\sum_{n=\pm1}^{\pm\infty}\left\{\frac{J_n[(m\omega_c + n\omega_v)\frac{M}{2}\frac{\pi}{\omega_c}]}{(m\omega_c + n\omega_v)\frac{\pi}{\omega_c}}[Sin\,[(m\omega_c + n\omega_v).(t - \frac{\pi k}{\omega_c}) - n\frac{\pi}{2}]\right.$$

$$\left. - Sin\,[(m\omega_c + n\omega_v).(t + \frac{\pi k}{\omega_c}) + n\frac{\pi}{2})]]\right\}$$

Eqn A.8.5.b

The first term's trigonometric part can be simplified since :

$$Sin\left(n\omega_v t - n\pi k\frac{\omega_v}{\omega_c} - n\frac{\pi}{2}\right) - Sin\left(n\omega_v t + n\pi k\frac{\omega_v}{\omega_c} + n\frac{\pi}{2}\right)$$
$$= -2\,Cos(n\omega_v t).\,Sin\left(n\pi k\frac{\omega_v}{\omega_c} + n\frac{\pi}{2}\right)$$

Eqn A.8.5.c

Similarly, the second term's trigonometric part can be simplified :

$$Sin(m\omega_c t - m\pi k) - Sin(m\omega_c t + m\pi k) = -2\,Cos(m\omega_c t).\,Sin(m\pi k)$$

Eqn A.8.5.d

and the last term can be simplified :

$$Sin\left[(m\omega_c + n\omega_v).\left(t - \frac{\pi k}{\omega_c}\right) - n\frac{\pi}{2}\right] - Sin\left[(m\omega_c + n\omega_v).\left(t + \frac{\pi k}{\omega_c}\right) + n\frac{\pi}{2}\right]$$
$$= -2\,Cos(m\omega_c + n\omega_v)t\,.\,Sin\left[(m\omega_c + n\omega_v).\frac{\pi k}{\omega_c} + n\frac{\pi}{2}\right]$$

Eqn A.8.5.e

and by putting equations A.8.5.c-e back in A.8.5.b, the spectrum for SYMPWM can be found (cf. equation 2.3.4.a) :

$$F_{SYM}(t) = k + \sum_{n=1}^{\infty}\left\{\frac{2J_n\left(n\frac{\omega_v}{\omega_c}\pi\frac{M}{2}\right)}{n\pi\frac{\omega_v}{\omega_c}}Cos(n\omega_v t).\,Sin\left(n\pi k\frac{\omega_v}{\omega_c} + \frac{n\pi}{2}\right)\right\}$$
$$+\sum_{m=1}^{\infty}\left\{\frac{2J_0\left(m\pi\frac{M}{2}\right)}{m\pi}Cos(m\omega_c t).\,Sin(m\pi k)\right\}$$
$$+\sum_{m=1}^{\infty}\left\{\sum_{n=\pm 1}^{\pm\infty}\left\{\frac{2J_n\left[\pi\frac{M}{2}\frac{(m\omega_c + n\omega_v)}{\omega_c}\right]}{\pi.\frac{(m\omega_c + n\omega_v)}{\omega_c}}Cos\left[(m\omega_c + n\omega_v)t\right]Sin\left[\frac{m\omega_c + n\omega_v}{\omega_c}\pi k + n\frac{\pi}{2}\right]\right\}\right\}$$

Eqn A.8.5.f

A.8.6 : Derivation of 2SCPWM's spectrum from Uniform TEPWM & LEPWMs'

In the last section the spectrum for uniformly sampled symmetric modulation was found by summing leading and trailing edged versions of the same sampled signal. If the same signal is sampled at twice the rate, this permits the sampled value from the start of the sample period to be used for the leading edged component of the pulse and that from the centre of the pulse period can modulate the trailing edged component. As before the derivation of such a modulation type (which is referred to as 2SCPWM) can be achieved by summing two non-overlapping spectra from appropriately scaled and delayed versions of the uniformly sampled trailing edged modulation spectrum. In this case, the leading edged component is required from a signal sampled half a pulse period in advance of the trailing edged component's sampling instants. By taking the spectrum of uniformly sampled leading edged modulation, replacing the time index, 't' by 't+π/ω_c' and halving the modulation index and offset, the contribution to the output spectrum can be found :

$$F(t) = \frac{k}{2} - \sum_{n=1}^{\infty} \left\{ \frac{J_n(n\pi \frac{M}{2} \frac{\omega_v}{\omega_c})}{n\pi \frac{\omega_v}{\omega_c}} . Sin(-n\omega_v t + n\omega_v \frac{\pi}{\omega_c} - n\pi k \frac{\omega_v}{\omega_c} - n\frac{\pi}{2}) \right\}$$
$$+ \sum_{m=1}^{\infty} \left\{ \frac{1}{m\pi} [Sin(-m\omega_c t + m\omega_c \frac{\pi}{\omega_c}) - J_0(m\frac{M}{2}\pi) . Sin(-m\omega_c t + m\omega_c \frac{\pi}{\omega_c} - m\pi k)] \right\}$$
$$- \sum_{m=1}^{\infty} \sum_{n=\pm 1}^{\pm\infty} \left\{ \frac{J_n[(m\omega_c + n\omega_v)\frac{\pi}{\omega_c}\frac{M}{2}]}{(m\omega_c + n\omega_v)\frac{\pi}{\omega_c}} . Sin[(m\omega_c + n\omega_v).(-t + \frac{\pi}{\omega_c} - \frac{\pi k}{\omega_c}) - n\frac{\pi}{2}] \right\}$$

Eqn A.8.6.a

To this, the spectral contribution from the trailing edged component can be added by taking equation A.8.5.a with its modulation index and offset halved :

$$F(t) = \frac{k}{2} - \sum_{n=1}^{\infty} \left\{ \frac{J_n(n\pi \frac{M}{2} \frac{\omega_v}{\omega_c})}{n\pi \frac{\omega_v}{\omega_c}} . Sin(n\omega_v t - n\pi k \frac{\omega_v}{\omega_c} - n\frac{\pi}{2}) \right\}$$
$$+ \sum_{m=1}^{\infty} \left\{ \frac{1}{m\pi} [Sin(m\omega_c t) - J_0(m\frac{M}{2}\pi) . Sin(m\omega_c t - m\pi k)] \right\}$$
$$- \sum_{m=1}^{\infty} \sum_{n=\pm 1}^{\pm\infty} \left\{ \frac{J_n[(m\omega_c + n\omega_v)\frac{\pi}{\omega_c}\frac{M}{2}]}{(m\omega_c + n\omega_v)\frac{\pi}{\omega_c}} . Sin[(m\omega_c + n\omega_v).(t - \frac{\pi k}{\omega_c}) - n\frac{\pi}{2}] \right\}$$

Eqn A.8.6.b

As in the symmetric modulation case the trigonometric parts of the three terms can be simplified by converting to the 'sum and difference' angle version, thus, for the first term :

$$Sin(n\omega_v t - n\pi k \frac{\omega_v}{\omega_c} - n\frac{\pi}{2}) + Sin(-n\omega_v t + n\omega_v \frac{\pi}{\omega_c} - n\pi k \frac{\omega_v}{\omega_c} - n\frac{\pi}{2})$$
$$= -2 Cos(n\omega_v t - n\frac{\pi}{2}\frac{\omega_v}{\omega_c}) . Sin[n\frac{\pi}{2}(1 + [2k-1].\frac{\omega_v}{\omega_c})]$$

Eqn A.8.6.c

similarly, for the second term :

$$Sin\,(m\omega_c t) + Sin\,(-m\omega_c t + m\omega_c \frac{\pi}{\omega_c}) = 2\,Cos\,(m\omega_c t - m\frac{\pi}{2}).\,Sin\,(m\frac{\pi}{2})$$

Eqn A.8.6.d

for the third term :

$$Sin\,(m\omega_c t - m\pi k) + Sin\,(-m\omega_c t + m\omega_c \frac{\pi}{\omega_c} - m\pi k) = 2\,Cos\,(m\omega_c t - m\frac{\pi}{2}).\,Sin\,(m\frac{\pi}{2} - m\pi k)$$

Eqn A.8.6.e

and for the last term :

$$Sin\,[(m\omega_c + n\omega_v).(-t + \frac{\pi}{\omega_c} - \frac{\pi k}{\omega_c}) - n\frac{\pi}{2}] + Sin\,[(m\omega_c + n\omega_v).(t - \frac{\pi k}{\omega_c}) - n\frac{\pi}{2}]$$

$$= -2\,Cos\,[(m\omega_c + n\omega_v).(t - \frac{\pi}{2}\omega_c)].\,Sin\,[\frac{m\omega_c + n\omega_v}{\omega_c}.\pi(k - \frac{1}{2}) - n\frac{\pi}{2}]$$

Eqn A.8.6.f

Hence in summary, the spectrum of 2SCPWM is found by replacing the trigonometric parts of either equation A.8.6.a or b with equations A.8.6.c-f, which yields (cf. equation 2.3.6.a) :

$$F_{2SC}(t) = k + \sum_{n=1}^{\infty}\left\{\frac{2J_n(n\pi\frac{M}{2}\frac{\omega_v}{\omega_c})}{n\pi\frac{\omega_v}{\omega_c}}\,Cos\,(n\omega_v t - n\frac{\pi}{2}\frac{\omega_v}{\omega_c})\,Sin\,(n\frac{\pi}{2}[(2k-1)\frac{\omega_v}{\omega_c} + 1])\right\}$$

$$+ \sum_{m=1}^{\infty}\left\{\frac{2}{m\pi}\,Cos\,(m\omega_c t - m\frac{\pi}{2}).\,[Sin\,(m\frac{\pi}{2}) + J_0(m\frac{M}{2}\pi)\,Sin\,([2k-1]m\frac{\pi}{2})]\right\}$$

$$+ \sum_{m=1}^{\infty}\left\{\sum_{n=\pm1}^{\infty}\left\{\frac{2J_n[\pi\frac{M}{2}\frac{(m\omega_c + n\omega_v)}{\omega_c}]}{\pi\frac{(m\omega_c + n\omega_v)}{\omega_c}}\,Cos\,[(m\omega_c + n\omega_v)(t - \omega_c\frac{\pi}{2})]\,Sin\,[\frac{\pi}{2}(2k-1)\frac{m\omega_c + n\omega_v}{\omega_c} - n\frac{\pi}{2}]\right\}\right\}$$

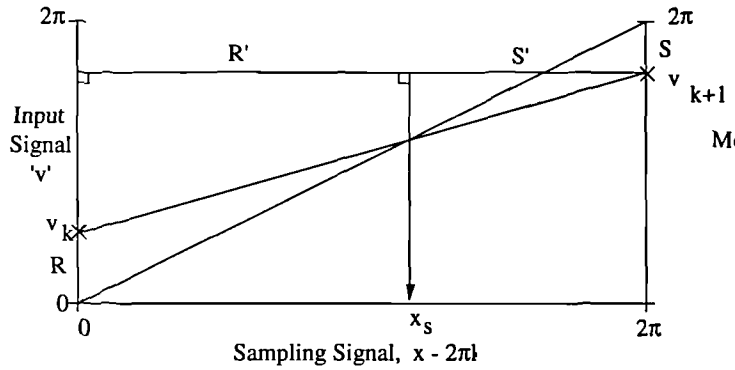
Eqn A.8.6.g

In this grouping of the trigonometric parts, special care has been taken to show the factor '(2k-1)' where it occurs, so that the reduction of even order harmonic and sideband terms associated with this modulation type can be seen. It should be noted that such a reduction in the output distortion will only occur when k=0.5, since if this is not the case, '(2k-1)' will be non-zero.
(cf. equation 2.3.6.a)

A.8.7 : Initial steps in the derivation of WAPWM's spectrum

'Weighted Average' or first order pseudo-natural pre-compensation for uniformly sampled trailing edged modulation can be analysed in a similar way to naturally sampled, trailing edged modulation using equation A.8.2.o; however, there is no simple substitution to complete its expression in terms of Bessel functions. With the assistance of numerical evaluation, the tonal performance can still be found but the expressions for that performance are no longer concise. The initial steps in this analysis are algebraic as before and are shown below. With this re-expression of the problem, numerical packages such as 'Reduce', 'Mathematica' or 'MathCad' can be used to arrive at specific results for given single tone input frequency and amplitude and for a given carrier frequency. Multiple numerical analysis of this type can the be used to tabulate results such as shown in figures 5.2.3.d & e.

As shown in chapter 5, each individual pre-compensated value can be calculated from its neighbouring pair of successively uniformly sampled values (cf. equation 5.2.3.c) :



Input Range : $0 < v < 2\pi$
Modulation Depth : $0 < M < 1$
DC Offset : $0 < k < 1$
Sampling Rate : $x = \omega_c t$
Signal Rate : $y = \omega_v t$
Input Signal : $v = B + QCos(y)$
where : $B = 2\pi k, Q = M\pi$

Figure A.8.7.a : Naming Conventions for Weighted Average Pre-compensation analysis.

Using the above nomenclature :

$$R' = 2\pi\frac{R}{R+S} \quad , \quad v_k = B + QCos(\frac{\omega_v}{\omega_c}.2\pi k) \quad , \quad v_{k+1} = B + QCos(\frac{\omega_v}{\omega_c}.2\pi[k+1])$$

hence :

$$x_s = 2\pi\left[\frac{v_k}{v_k + (2\pi - v_{k+1})}\right] = \frac{v_k}{1 - \frac{v_{k+1} - v_k}{2\pi}}$$

$$= \frac{B + QCos(\frac{\omega_v}{\omega_c}.2\pi k)}{1 - \frac{Q}{2\pi}.\{Cos(\frac{\omega_v}{\omega_c}.2\pi[k+1]) - Cos(\frac{\omega_v}{\omega_c}.2\pi k)\}}$$

Eqn A.8.7.a

and in the k-th interval, $x_k - 2\pi k = x_s$.

Appendix 9 : A DSP Case Study using Motorola's DSP56004FJ40.

A.9.1 : Design and Implementation of a Zero Interleaved Noise Shaper.

As shown in chapter four, two sided modulation types such as DSPWM and 2SCPWM require independent noise shaping of the data modulating each edge. Interleaving the noise shaper's feedback filter with zero valued coefficients provides the independence and this can be implemented using delay elements of two clock cycles duration. An example assembly subroutine in presented in section A.9.5 using such noise shaping with a 7 th order NTF before zero interleaving, wordlength reduction from 24 bits to 8 in a ten times interpolated data stream (441000 kHz), while taking into account the restrictions of Motorola's DSP56004. The NTF itself was designed for minimum gain by optimisation with targets defined by equation 4.3.3.i, and its theoretical frequency response is shown below:

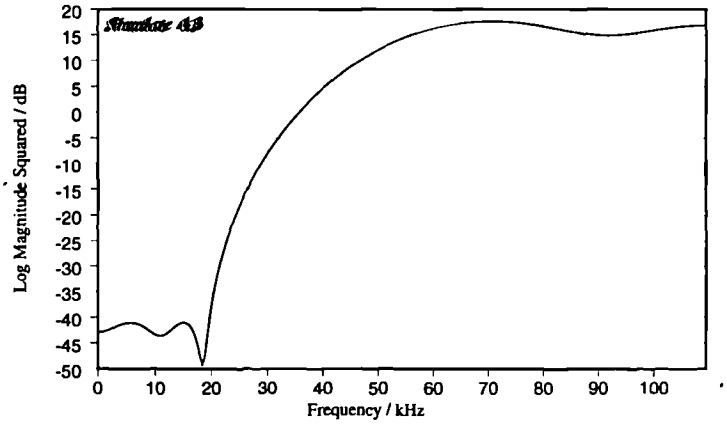


Figure A.9.1.a : Frequency Response of the Chosen Noise Transfer Function

Requantisation of the signal during the processing of the noise shaper can be achieved by logical masking or scaling down of the signal so that it is truncated to the register wordlength. Truncation of the signal by scaling is useful because this ensures the sign bit extension data is performed without overhead and external scaling can be hard-wired without further glue logic. However, this creates problems in finding the error since subtraction across the quantiser cannot be performed without re-scaling the output. To solve this, this error and output can be found separately: quantisation of the output can be achieved by scaling and truncation, the error can be found by a bit-wise AND to mask off the MSBs. The revised structure for achieving this is shown below :

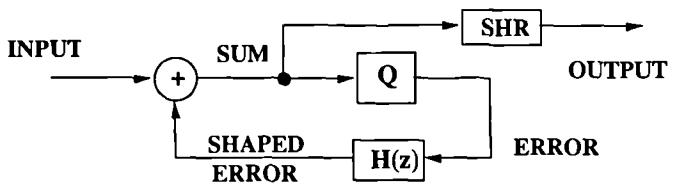


Figure A.9.1.b : Noise Shaping with Separate Error and Output Evaluation

Since truncation produces a result which is always smaller than the input for numbers in a two's complement notation, the sign of the error is positive. By loading an accumulator with the mask and AND-ing the data to it (rather than the other way round), the nulled MSBs of the mask (required to

remove the MSBs of the data which becomes the error) are sign bit-extended automatically, setting the sign of the AND result to be positive. It is this sign bit extension of the processor which causes the unusual behaviour : a non-commutative AND operator!

The largest coefficient for this NTF is C_2 (=3.147) but the processor uses fractional mathematics (-1.0 < data < +0.9999997) so the coefficients have to be stored in a scaled down form. To avoid additional scaling of the error during execution, bus alignment between the accumulators and registers can be set to introduce one arithmetic shift left. A scale-up factor of four can be achieved by moving data across this interface twice for each time the feedback loop is evaluated, so storing the coefficients at 1/4 scale can be accommodated. A block diagram of this structure is shown below :

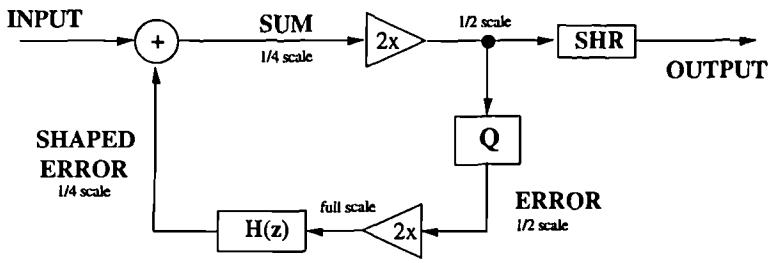


Figure A.9.1.c : Revised Noise Shaper Structure to Permit Coefficient Storage at 1/4 Scale

The filter itself is an FIR design (to avoid near signal band noise as discussed in section 4.5) and this would normally be evaluated by repeatedly multiplying-and-accumulating (MAC) each delayed error with the appropriate coefficient. The processor supports indirect addressing using address pointers which can be incremented after use so as to point to the next required data, and parallel moves on two data busses (one for the delayed error, say 'x', one for the coefficients, say 'y'). By forcing the modulus of the delayed error buffer to be the same as the length of the filter, a circular buffer can be created whose pointer processes by one in each loop evaluation to implement a clocking action for the delayed error signal. The newest error signal is written over the oldest in each cycle and the pointer is then not incremented so as to assure the required stepping action, backward through the buffer.

This filter memory management is very efficient but requires special care for reduced complexity filters (see section 4.5) if the complexity reduction is to be retained. For the frequency response shown in figure A.9.1.a, the associated impulse response has been reduced to have only five non-zero taps as shown below :

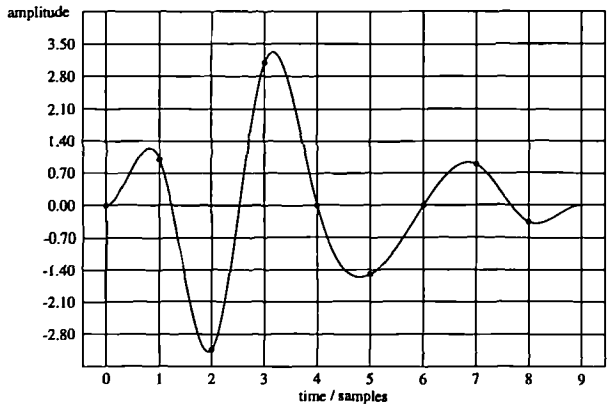


Figure A.9.1.d : The Impulse Response of An Example of a Reduced Complexity NTF

For this particular filter, coefficients C_3 and C_5 need not be evaluated, however the associated error values cannot be forgotten because they will be required in earlier and later filter evaluations. To overcome this a revised circular buffer has to be used for the error, such that the pointer can be incremented by a fixed amount, but skips the error data which is not currently required and starts and finishes the filter evaluation so that the newest error can be stored over the oldest. A scheme to do this is presented below (figure A.9.1.e). In this diagram, the circular buffer has been unravelled and duplicated so that the step size can clearly bee seen. In reality, every occurrence of each piece of data is the same location in memory.

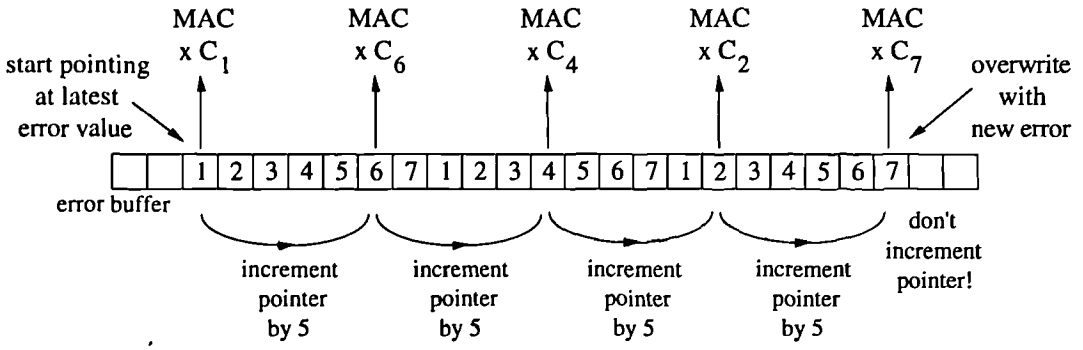


Figure A.9.1.e : Data Hopping to Permit Fast Evaluation of Reduced Complexity Filters

As each delay error datum is retrieved, the coefficient with which to multiply it has to be retrieved from a separate buffer. This can be accessed in the conventional way provided the coefficients are stored in the appropriate order (ie.: C_1, C_6, C_4, C_2, C_7). For zero interleaved operation, where two channels of delayed error are to be filtered independently, the circular buffer is simply enlarged to modulo 14 instead of 7 and the pointer increment is doubled from 5 to 10. At the end of processing each sample, the pointer should be incremented by one to point to the alternate set of data (which was aligned with the zero coefficients in evaluating the filter for sample one). For stereo operation, two error buffers are required, although the same pointer can be used if the buffers reside in different memory pages.

A.9.2 : Dither Generation On-Board a Digital Signal Processor.

Several techniques exist for the generation of dither, all using pseudo random noise sources. Since triangular PDF, high pass dither is recommended as the best to audibly decorrelate quantisation effects [VAN89], differencing a rectangular PDF random number generator is favoured. Three families of random numbers generators are suitable for implementation on a DSP, those based in maximal length sequences, those based on linear congruential number generators and those based on chaotic systems such as a 'tent map' or the 'logistic equation'. Quantised chaotic generators usually have a shorter period that the maximal length sequence or linear congruential number generators can provide, and only produce rectangular PDF under special conditions; the maximal length sequence requires the use of bit-wise operators that are not efficiently implemented on DSPs. Hence, a linear congruential number generator is favoured here.

The basic algorithm for a linear congruential number generator is captured in equation A.9.2.a:

$$X_{N+1} = |k.X_N + c|_M \qquad \text{Equation A.9.2.a}$$

The modulus, 'M', can be chosen for two's complement systems to be 2^B, where 'B' is the available wordlength (in bits) of the register in the DSP. This permits efficient evaluation of the modulus by calculating the product of the last random number in a larger accumulator and then taking the lower portion of it as the new random number. This does incur a small amount of overhead, since the offset has to be pre-scaled to be in the bottom portion of the accumulator, and the new random number cannot be removed from the accumulator while other parallel moves are taking place (this is a 56004-specific limitation). A section of DSP56004 assembly code for this is presented below :

```
sdml            equ     $00ff           ;; dither seed/multiplier, x:seed, y:multiplier
ofst            equ     $00fe           ;; dither 48 bit constant, x:sign, y:constant
ditr            equ     $00fd           ;; high pass filtered dither
                move    #>$abcdef,x0                ;; seed dither != 0
                move    x0,x:sdml
                move    #>$0006cd,y0                ;; define multiplier = 1741
                move    y0,y:sdml
                move    #>$000000,x0                ;; constant sign is irrelevant (force +ve for Bl use)
                move    x0,x:ofst
                move    #>$3c6eed,y0                ;; define constant value = 3960557
                move    y0,y:ofst
                move    #>$000000,x0                ;; clear hpf'd dither's MS word
                move    x0,x:ditr
;; main program
loop            move            l:sdml,x            ;; load dither seed and multiplier
                mpy     x0,x1,a l:ofst,b            ;; produce mX, load constant with nulled MS word
                addr    b,a     x:sdml,b0           ;; counteract shl in mpy using fractional maths
                sub     b,a     a0,x:sdml           ;; produce hpf dither, move out seed
                asr     a
                move    a0,y:ditr                   ;; write out the dither
                jmp     loop                        ;;
                end                                 ;; end of program
```

Figure A.9.2.a : Assembly Code for a Triangular PDF Dither Source

Suitable values for the multiplier and offset for modulo 2^{24} are recommended by Knuth as k=1741, and c=3960557 or 3960581 (nearest primes to $(\sqrt{5}-2).2^{24}$). The sequence length for these variables is in fact 4194303 samples, which lasts 9.5 seconds at 441 kHz. This could be modified to last longer with slightly more processing overhead. Subsets of the wordlength can be used for separate dither samples if more coarsely quantised dither can be tolerated (eg. 8 bit quantised dither would allow 3 dither samples to be taken from each 24 bit random number). For stereo signals, separate dither sources are required for each channel to avoid the dither being located centrally in the stereo image; this can be achieved by simply initialising the random number generators for each channel to well separated parts of the sequence cycle.

A typical spectrum (figure A.9.2.b) and PDF (figure A.9.2.c) are shown below. These signals has been taken from the processor after high pass filtering (by differencing) which also constructs the triangular PDF, thus the high pass spectral and triangular PDF properties are visible.

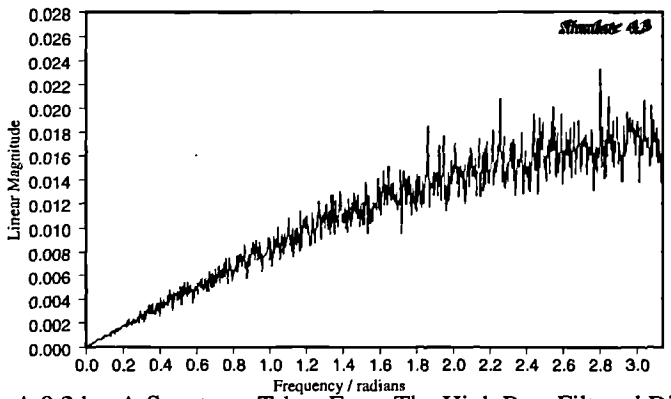


Figure A.9.2.b : A Spectrum Taken From The High Pass Filtered Dither Source.

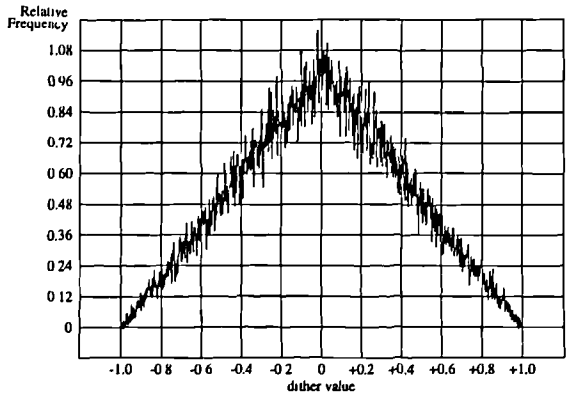


Figure A.9.2.c : A Plot of PDF Taken From The High Pass Filtered Dither Source.

A.9.3 : Psychoacoustic Noise Shaping for Maximum Subjective Performance.

Since noise components of a flat floor can be more easily detected in the 3-4 kHz region and less easily detected near 20 kHz, subjective performance of a noise shaper designed for maximum SNR can be improved by designing requantisation noise to follow the inverse of a human audibility curve. Requantisation noise has already been designed to be at as low a level as possible, so these modifications will only affect noise at or near the heading threshold, so the E-weighting is the most suitable audibility curve to design with (this is measured at the 15 phon level).

As discussed in chapter four, the log-integral of a noise transfer function can, at best, be zero and is usually more than zero. Thus if the NTF stopband is over-designed, the power gain will be forced to increase ... possibly to a level at which noise will be re-modulated by the PWM to recur in the audio band. Thus a trade-off can be anticipated between how deep the stopband requantisation noise is suppressed and how much the consequences of this might have repercussions later. If the stopband is deeper than the inverse E-weighted curve would suggest in frequency bands to which the listener is insensitive, this can be considered an over-design and such effort can be better used in a sensitive band.

By designing a set of filters with increasing suppression but similarly shaped inverse E-weighted stopband floors, and then measuring the E-weighted SNR after modulation, the near optimal subjective performance can be found. A summary of such a set of filters is shown below, after enforcing the reduced complexity constraints of the NTF presented in section A.9.1:

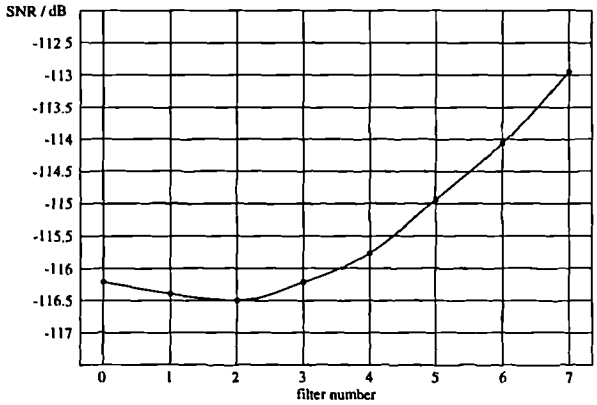


Figure A.9.3.a ; A Graph of Subjective SNR After Modulation for a Family of NTFs

This graph shows that the filters with greater stopband suppression start to reduce the output SNR after modulation, and the filters with lower stopband suppression simply have higher in-band noise, thus an optimum exists : filter 2. A spectrum of its output after modulation is shown below :

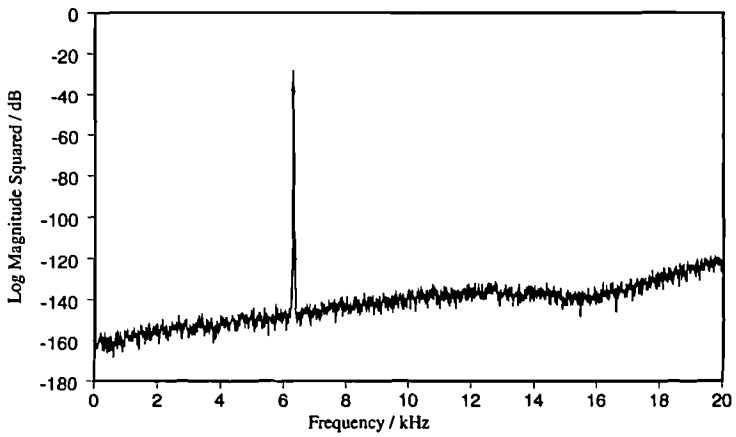


Figure A.9.3.b : PWM Output Spectrum using a Psychoacoustically Optimal Noise Shaper.

The performance of this filter becomes of particular interest in a system in which some volume control is implemented within DSP. As the input signal is scaled down in DSP, the input quantisation noise is also scaled down, making the requantisation noise more significant and hence increasing the need for a larger dynamic range in the converter (although this is achieved subjectively). In the limit the converter is limited by unwanted circuit non-idealities rather than the PWM or DSP so a compromise which yields better (flat) SNR measurements may be preferable if the advantages of greater dynamic range are going to be lost through some other route.

A.9.4 : Fast Evaluation of Enhanced Sampling Pre-compensation.

As discussed in chapter five, pre-compensation is required for reduction of the baseband harmonic distortion arising from uniformly sampled modulation types. One of particular interest is enhanced sampling because of its simple evaluation and effectiveness for reduction of odd order harmonic components. When combined with 2SCPWM (which has no inherent even order components) a near distortion free system can be constructed.

Enhanced sampling used a combination of uniformly and naturally sampled data values to form the new data value but a first order approximation to the naturally sampled value will suffice for harmonic minimisation providing an appropriate value for the sampling factor 'ε' is used. Finding the pre-compensated value requires knowledge of the system in which the data will be used since offsets and scales have to be taken into account before pre-compensation. The pre-compensation itself is non-linear, so linear modifications such as scaling and offsets become inappropriately compensated for if applied between the enhanced sampler and the pulse width modulator.

The two edges of each pulse used in a two sample system have to be pre-compensated separately (the naturally sampled data would have come from different parts of the sampling wave) so two sets of equations have to be used in the evaluation of the pre-compensation. These will be presented using the following diagram (figure A.9.4.a) which shown both edges of one pulse's period and the sampling signals and guard band intervals for it.

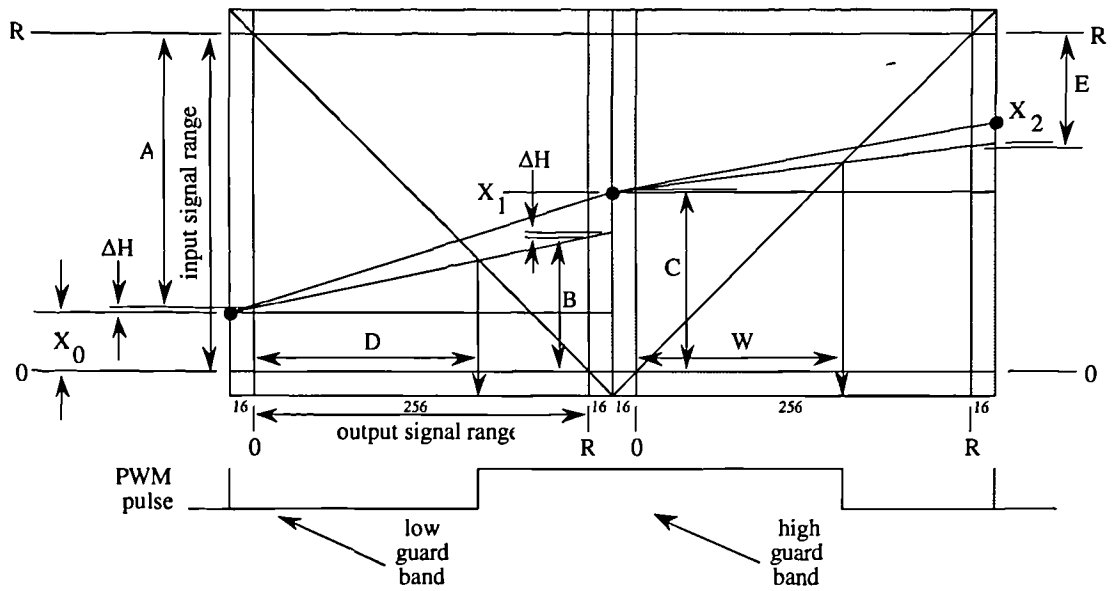


Figure A.9.4.a : Pre-compensation Geometry For an Enhanced Sampled Two Sided Pulse

By similar triangles for the trailing edge of the pulse :

$$\frac{C}{W} = \frac{E}{R - W}$$ Equation A.9.4.a

where :

$$C = X_1 + \Delta H \quad \text{and} \quad E = R - X_2 + \Delta H - \varepsilon.(X_2 - X_1)$$ Equation A.9.4.b

hence :

$$W = \frac{X_1 + \Delta H}{R - [X_2 - \varepsilon.(X_2 - X_1)] + \Delta H + X_1 + \Delta H}$$ Equation A.9.4.c

which after collating terms and substituting $\Delta H = 16/288.(1-\varepsilon).(X_2-X_1)$ yields :

$$W = \frac{(X_1 + \Delta H)/R}{1 - \partial}$$ Equation A.9.4.d

where :

$$\partial = (\frac{1 - \varepsilon}{R}).(\frac{256}{288}).(X_2 - X_1)$$ Equation A.9.4.e

For small '∂' , equation A.9.4.d can be re-expressed as :

$$W = (X_1 + \Delta H)/R . \sum_{k=0}^{\infty} \{\partial^k\}$$ Equation A.9.4.f

In this equation, the approximate value of the summation of the series can be found by taking a limited number of terms. For a full amplitude sinusoid at the maximum frequency which produces an odd harmonic below 20 kHz (6.6 kHz), in a 441 kHz sampled system, the worst case value of (X_2-X_1) is 9.5% of full scale, so with ε=0.36875 (see figure 5.4.2.a) in units of 'R', the worst case ∂ is :

$$\partial_{max.} = \frac{1 - 0.36875}{1} . \frac{256}{288} . 0.095 = 0.0534$$ Equation A.9.4.g

Assuming a 16 bit source, and requiring that the error be smaller than half an LSB, the required number of terms, K_{min} will be :

$$K_{min.} = \text{Integer}[\ \text{Log}(2^{-16}) / \text{Log}(\partial_{max})\ ] = 3 \qquad \text{Equation A.9.4.h}$$

Re-expressing the series using these terms alone, and grouping similar signed data to minimise truncation effects in calculation, the ESPWM algorithm can be defined as :

$$W = (X_1 + \Delta H).[(1+\partial^2)+\partial.(1+\partial^2)] \qquad \text{Equation A.9.4.h}$$

In this equation, various other factorisations have been considered but none is as computationally efficient on this DSP where multiply-and-accumulate instructions can be completed in one instruction cycle. When combined with the processor's capability to complete parallel moves with no processing overhead and the limited number of registers that can be used, this algorithm proved to be the best by 3 instruction cycles over its nearest rival. Noting that ΔH is small compared to X_1 , this can be simplified further to be :

$$W = X_1 + \partial.(\partial X_1) + \partial.(X_1 + \partial.(\partial X_1)) \qquad \text{Equation A.9.4.i}$$

Similarly, the delay can be found to be approximately :

$$D = (1 - X_0) + \partial.(\partial.(1 - X_0)) - \partial.((1 - X_0) + \partial.(\partial.(1 - X_0))) \qquad \text{Equation A.9.4.j}$$

and from the delay, the leading edge component of the width can be calculated as :

$$LEC_{width} = 1 - D \qquad \text{Equation A.9.4.k}$$

It is important that the guard bands are so chosen that the output scale is a power of two. This permits the addition of guard bands after quantisation by truncation in the noise shaper and is why the guard bands are allowed for, rather than included, in the calculation of the pre-compensation.

This algorithm has been tested using bit-true simulation and hardware and an example spectrum after modulation is shown below (figure A.9.4.b). In this spectrum, the third harmonic would have been at a level of -104.5 dBFS if a floating point division had been used instead of the numerically simplified algorithm presented above (which produces a 3 rd harmonic at -103.2 dBFS). This difference, of just over one decibel, is considered to be worthwhile considering the large amount of simplification that has been achieved. Further tests with more complex tonal groups show even less deviation from the best that can be achieved with this pre-compensation strategy.

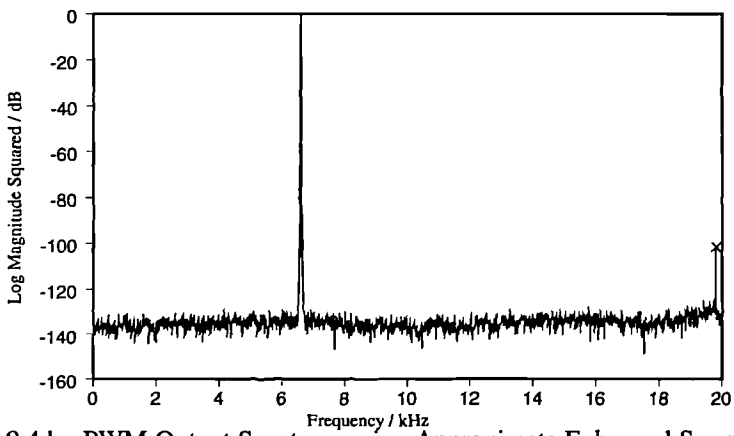


Figure A.9.4.b : PWM Output Spectrum using Approximate Enhanced Sampling & ZINS.

A.9.5 : Assembly Code for a 440 kHz ZINS & ESPWM Algorithm.

By building together the code for approximate ESPWM pre-compensation, reduced complexity ZINS, and adding polling for input and output timing, the following programme was developed and run at 441 kHz in a real time hardware implementation.

```
        page    132                     ; TST 015.ASM: 108 lpp (3rd order ES @ 1/2 scale & 7th order NS @ 1/4 scale,fixed volume)
                                        ;                      ( Timing by polling, quantisation by truncation, without dither. )
;       - ------------------------------------------------------------------------------------------------------------------------;
; Title   : 5 tap logical MFNS, 3 term delay/width ESPWM, with software sync. for PWM5G & 56004 DSP prototype boards             ;
; Author  : Rod. Hiorns                                                                                                         ;
; C mpany : B&W (Loudspeakers), Elm Grove Lane, Steyning, West Sussex, BN44 3SA, UK.                                            ;
; Version : Motorola DSP56000 Assembler Version 5.0.1 (1992)                                                                    ;
;         : Motorola DSP56004 User Interface Program 4.1.2 (1993)                                                               ;
; Date    : 03/05/94                                                                                                            ;
;             - -  --     - - ----- ------------------------------------------------------------------------------------------;

        opt     cc                      ; count instruction cyles
        opt     mu                      ; compile memory map

; set i   and other addresses (peripherals are x memory mapped!)

brc             equ     $ffe0           ; SAI baud rate control register
r s             equ     $ffe1           ;     receiver control/status register
rx              equ     $ffe2           ;     data receive register 0
rx1             equ     $ffe3           ;     data receive register 1
t s             equ     $ffe4           ;     transmitter control/status register
tx              equ     $ffe5           ;     data transmit register 0
tx1             equ     $ffe6           ;     data transmit register 1
tx2             equ     $ffe7           ;     data transmit register 2

ebar0           equ     $ffe8           ; EMI base address register 0
e r             equ     $ffe9           ;     offset register 0
edr             equ     $ffea           ;     data register 0
e  r            equ     $ffeb           ;     control status register
ebar1           equ     $ffec           ;     base address register 1
eor1            equ     $ffed           ;     offset register 1
edr1            equ     $ffee           ;     data register 1
ercr            equ     $ffef           ;     refresh control register

h kr            equ     $fff0           ; SHI clock control register
h sr            equ     $fff1           ;     control status register
h ar            equ     $fff2           ;     I2C slave address register
hrtx            equ     $fff3           ;     host receive transmit register

ewor            equ     $fff6           ; EMI write offset register
gpi r           equ     $fff7           ; GPIO control data register
  dbr           equ     $fffc           ; GDB register (ONCE port)
pll r           equ     $fffd           ; PLL control register
ipr             equ     $ffff           ; Interrupt priority register

rldf            equ     14              ; receiver "left" data full bit No (decimal)
rrdf            equ     15              ; receiver "right" data full bit No (decimal)

re et           equ     $    0          ; hardware reset vector
 tart           equ     $0040           ; start of program

. define l cal program parameters

 rd             equ     $0d             ; NTF modulus (= 2.[NTF order]-1 ... max.16!)
tap             equ     $04             ; NSF modulus (= [NSF length]-1 ... max.16!)
gap             equ     $0a             ; NSF increment to avoid zero-valued coeffs
ram             equ     $0 00           ; Y-RAM base for program storage
lde             equ     $0000           ; X RAM base for left delayed error
rde             equ     $0010           ; X RAM base for right delayed error
hn              equ     $0010           ; Y RAM base for NS coeffs
log             equ     $0020           ; X&Y-RAM base for logical & scalar constants

. define hardware /RESET response

                org     PI:reset,PI:reset               ;0
                jmp     start                           ;4  RESET vector, P:$0000

                org     PI:start,PI:start               ;0 ------------------------------------------------------------------;

; define program specific addresses

                move    #rx0,r0                         ;4 Rx register addresses' location in X-RAM
                move    #lde,r1                         ;2 left error pointer
                m vec   #ord,m1                         ;2 left error circular buffer modulo
                move    #gap,n1                         ;2 left error circular buffer gap size
                move    #rde,r2                         ;2 right error pointer
                movec   #ord,m2                         ;2 right error circular buffer modulo
                move    #gap,n2                         ;2 right error circular buffer gap size
                move    #log,r3                         ;2 logical & scalar constants' location in X&Y-RAM
                movec   #>$5,m3                         ;4 logical & scalar constants' circular buffer modulo
                move    #>$2,n3                         ;4 logical & scalar constants' circular buffer step size
                move    #tx0,r4                         ;4 Tx register base address location in X-RAM
                move    #ram,r5                         ;2 temporary storage pointer
                movec   #>$5,m5                         ;4 temporary storage circular buffer modulo
                move    #>$2,n5                         ;4 temporary storage circular buffer step size
                move    #hns,r6                         ;2 hns coeffs pointer location in Y-RAM
                movec   #tap,m6                         ;2 hns coeffs offset modulo
                move    #log,r7                         ;2 logical & scalar constants' location in Y-RAM only
                movec   #>$3,m7                         ;4 logical & scalar constants' circular buffer modulo
                move    #>$2,n7                         ;4 logical & scalar constants' circular buffer step size
```

```
; set up program specific values

                move            #>$47d27d,x0                ;4 256/288.(1-ε)/R ... (L&R,1&2) [ 47d27dH = 0.561111D ] 0:U/0.88:WA
                move                        #>$000100,y0    ;4 o/p scale ... (L&R,D&W) [ for n   16D, 1/2^n = 100H]
                move            x0,x:(r3)+  y0,y:(r7)+      ;2 ... stored in XY:0020
                move            #>$400000,x0                ;4 1/2 FS ... (L&R,1) [ 400000H = +1/2D ]
                move                        #>$007fff,y0    ;4 AND mask for 8 bit quantisation @ 1/2 scale (L&R) [007fffH]
                move            x0,x:(r3)+  y0,y:(r7)-      ;2 ... stored in XY:0021, r7 set for Y:0020 access
                move                        #>$00000a,y0    ;4 delay offset ... (L&R,1) [ 00000aH = 16 6D ]
                move                        y0,y:(r3)+      ;2 ... stored in Y:0022
                move                        #>$000123,y0    ;4 width offset ... (L&R,1) [ 000144H = 256+(16 6)+32 6D ]
                move                        y0,y:(r3)+      ;2 ... stored in Y:0023
                move                        #>$400000,y1    ;4 i/p offset (L&R,1&2) & delay scalar (L&R,1) ... [ 1/2D = 400000H ]
                tfr     y1,A                y1,y:(r3)+      ;2 ... preloaded in A and stored in Y:0023, later prefetched to x0
                move                        #>$3d70a4,y0    ;4 i/p scale (L&R,1&2) ... [ "max.vol!"   3d70a4H = 0.48D ]
                tfr     A,B     A,x0        y0,y:(r3)+n3    ;2 ... stored in Y:0024, prefetched in y0, r3 set for X:0021 access

; set up filter specific values (21 bit coefficients, stored @ " scale in re-arranged order)
; filter one : flat AB floor

                move                        #>$6457fc,y1    ;4 Hns(1) = dec 3.13574028  [NSI.0.B&W]
                move                        y1,y:(r6)+      ;2 store Hns(1)                 "
                move                        #>$e346d5,y1    ;4 Hns(6) = dec -0.897603035    "
                move                        y1,y:(r6)+      ;2 store Hns(6)                 "
                move                        #>$2fe269,y1    ;4 Hns(4) = dec 1.49638748      "
                move                        y1,y:(r6)+      ;2 store Hns(4)                 "
                move                        #>$9cbe57,y1    ;4 Hns(2) = dec -3.10176468     "
                move                        y1,y:(r6)+      ;2 store Hns(2)                 "
                move                        #>$0b857b,y1    ;4 Hns(7) = dec 0.360044479     "
                move                        y1,y:(r6)+      ;2 store Hns(7)                 "

; filter two : psychacoustic optimal floor

;               move                        #>$6d5e4b,y1    ;4 Hns(1) = dec 3.14776049 [NSI.EW.0.B&W]
;               move                        y1,y:(r6)+      ;2 store Hns(1)                 "
;               move                        #>$db11f7,y1    ;4 Hns(6) = dec -1.154056549    "
;               move                        y1,y:(r6)+      ;2 store Hns(6)                 "
;               move                        #>$3bee86,y1    ;4 Hns(4) = dec 1.87286675      "
;               move                        y1,y:(r6)+      ;2 store Hns(4)                 "
;               move                        #>$8ce7fc,y1    ;4 Hns(2) = dec -3.59668088     "
;               move                        y1,y:(r6)+      ;2 store Hns(2)                 "
;               move                        #>$0eb53b,y1    ;4 Hns(7) = dec 0.459623098     "
;               move                        y1,y:(r6)+      ;2 store Hns(7)                 "

; setup processor configuration registers

                movec   #$0300,sr                           ;4 mask all peripheral interrupts, /IRQA & /IRQB
                movep   #>$0,x:gpior                        ;6 deactivate GPIO port
                movep   #>$90,x:rcs                         ;6 personal reset (Rx)
                movep   #>$18,x:tcs                         ;6 personal reset (Tx)
                movep   #$6e0001,x:pllcr                    ;6 activate PLL - multiply by two
                movep   #>$101,x:brc                        ;6 baud rate (SCk=1 4MCk), used for Tx, same as Rx f r sync. circuit
                movep   #>$93,x:rcs                         ;6 Rx config: SM5842AP (2 i p, 24 bit, slave, +edge, MSB 1st, No INTs)

; define SAI timing by software polling (+cleanup)

time_set        jclr    #rldf,x:rcs,time_set                ;6 poll for left data ready (enabling Tx starts left data transfer)
                rep     #35                                 ;4 ( OK between 21 & 50 )
                move                        A,y:(r5)+       ;2 clear RAM & skip Rx Tx R W and Tx W W polling variati n x.rcs
                movep   #>$1B,x:tcs                         ;6 Tx config: PWM5G (2 o p, 24 bit, master, +edge, MSB 1st, No INTs)
                move            x:(r0)+,x1                  ;2 clear receiver backlog
                move            x:(r0)-,x1                  ;2 clear receiver backlog

; main program (Ts=192 cpu cycles = 96 instructions, nom., 48 per sample, 3p1+15+20 & 3p2+13+2+18+2 used)

loop_1          jclr    #rrdf,x:rcs,loop_1                  ;6 wait until first datum's ready in Rx register (WCk g es high)

;loop_1         movep   #>$100,x:gpior                      ;D6

                move            x:(r0)+,x1                  ;2                      read Rx0 (L)
                macr    +y0,x1,B x:(r0)-,x1  y:(r5)+,y1     ;2 SMC i/p (L),         read Rx0 (R),     prefetch x0 (L)
                macr    +y0,x1,A x:(r3)-,B   B,y:(r5)+      ;2 SMC i/p (R),         preload FS/2 (L), store x1 (L)
                macr    -x0,y1,B x:(r3)+n3,x1 A,y:(r5)-     ;2 find X (L),          prefetch (1-ε)/R, st re x1 (R)
                mpy     +y1,x1,A B,x:(r3)-   y:(r5)+n5,y1   ;2 find (1-ε)x0 (L),    store X (L),      prefetch x1 (L)
                macr    -y1,x1,A x:(r3)+,B   y:(r5),y1      ;2 find ∂ (L),          preload Fs 2 (R), prefetch x0 (R)
                macr    -x0,y1,B x:(r3)+,x0  A,y:(r5)-      ;2 find X (R),          prefetch X (L),   store ∂ (L) over x0 (R)
                mpy     +y1,x1,A B,x:(r3)    y:(r5)+,y1     ;2 find (1-ε)x0 (R),    store X (R),      prefetch x1 (R)
                macr    -y1,x1,A x:(r3)-,x1  y:(r5)+n5,y0   ;2 find ∂ (R),          prefetch X (R),   prefetch ∂ (L)
                mpyr    +y0,x0,A A,x0                       ;2 find ∂X (L),         save ∂ (R)
                mpyr    +x1,x0,B x:(r3)+,A   A,y1           ;2 find ∂X (R),         preload X (L),    save ∂X (L)
                macr    +y1,y0,A x:(r3)+,B   B,y1           ;2 find X+∂²X (L),      preload X (R),    save ∂X (R)
                macr    +x0,y1,B A,x1        y:(r5)+n5,y1   ;2 find X+∂²X (R),      save X+∂²X (L)    dummy read for postinc
                mac     +x1,y0,A B,x1        y:(r5)+n5,y1   ;2 find X+∂²X+∂(X+∂²X) (L), save X+∂²X (R), prefetch x1 (L)
                mac     +x1,x0,B x:(r1)+n1,x0 y:(r6)+,y0    ;2 find X+∂²X+∂(X+∂²X) (R), prefetch e1 (L), Hns(1)
                ori      #$08,mr                            ;2 set up one asl on accumulator exits
                mac     +y0,x0,A x:(r2)+n2,x0 y1,y:(r5)+n5  ;2 1st filter tap (L),      "       (R), dummy write (on ∂) for p stinc
                mac     +y0,x0,B x:(r1)+n1,x0 y:(r6)+,y0    ;2       "             (R), prefetch e6 (L), Hns(6)
                mac     +y0,x0,A x:(r2)+n2,x0 y1,y:(r5)-    ;2 2nd                 (L),      "       (R), store x1 (L)
                mac     +y0,x0,B x:(r1)+n1,x0 y:(r6)+,y0    ;2       "             (R), prefetch e4 (L), Hns(4)
                mac     +y0,x0,A x:(r2)+n2,x0               ;2 3rd                 (L),      "       (R)
                mac     +y0,x0,B x:(r1)+n1,x0 y:(r6)+,y0    ;2       "             (R), prefetch e2 (L), Hns(2)
                mac     +y0,x0,A x:(r2)+n2,x0               ;2 4th                 (L),      "       (R)
                mac     +y0,x0,B x:(r1),x0    y:(r6)+,y0    ;2       "             (R), prefetch e7 (L) [no postinc], Hns(7)
                macr    +y0,x0,A x:(r2),x0    y:(r7)+,y1    ;2 last                (L),      "       (R) [     "     ], prefetch o/p scale
                macr    +y0,x0,B A,x0         y:(r7),A      ;2       "             (R), save sum1 (L), preload quantiser mask (L)
                and      x0,A     B,x1         y:(r7)+,B     ;2 find error (L), save sum1 (R), preload quantiser mask (R)
                and      x1,B     A,x:(r1)+    y:(r7),A      ;2 find error (R), store error (L) [postinc], preload delay offset (L)
                mac     +x0,y1,A B,x:(r2)+    y:(r7)+,B     ;2 find delay (L), store error (R) [postinc], preload delay offset (R)
                andi     #$f7,mr                            ;2 clear scaling
                mac     +y1,x1,B A,x0         y:(r7)+,A     ;2 find delay (R),      save Q(delay) (L),    preload width offset (L)
                tfr      A,B      B,x1         y:(r3)+,y1    ;2 copy width offset,   save Q(delay) (R),    prefetch input offset (L)
                sub      x0,A     x0,x:(r4)+                 ;2 find width LHS (L), write tx0: delay (L)
                sub      x1,B     x1,x:(r4)-   y:(r3)+,y0    ;2 find width LHS (R), write tx1: delay (R), prefetch input scale (B)
                tfr      y1,A                  A,y:(r5)+n5   ;2 preload input offset,                     store width LHS (L)
```

```
loop 2          jclr    #rldf,x:rcs,loop 2              ;6 wait until second datum's ready in Rx register (WCk goes low)

;               movep   #>$101,x:gpior                  ;D6

                tfr      A,B     x:(r0)+,x1   B,y:(r5)-  ;2 copy i/p offset,    read Rx0 (L),     store width LHS (R)
                macr    +y0,x1,A x:(r0) ,x1   y:(r5)+n5,y1 ;2 SMC -i/p (L),     read Rx0 (R),     prefetch x0 (L)
                macr    +y0,x1,B x:(r3)+n3,x1 A,y:(r5)  ;2 SMC -i/p (R),     prefetch (1-ε)/R, store x1 (L)
                mpy      y1,x1,A                y:(r5)+,y0 ;2 find -(1-ε)x0 (L),                  prefetch x1 (L)
                macr    +x1,y0,A B,x0           y:(r5)+,y0 ;2 find ∂ (L),         save x1 (R)     prefetch x0 (R)
                mpy      x1,y0,B A,x:(r3)       B,y:(r5)+n5 ;2 find -(1-ε)x0 (R), store ∂ (L),    store x1 (R)
                macr    +x1,x0,B x:(r3),x0      y:(r5)+n5,A ;2 find ∂ (R),        prefetch ∂ (L), preload x0 (L)
                mpyr    +x0,y1,B B,x1           y:(r5)+,y1  ;2 find ∂X (L),       save ∂ (R),     dummy read for postinc
                mpyr    +x1,y0,B                B,y0        ;2 find ∂X (R),                       save ∂X (L)
                macr    +x0,y0,A B,x0           y:(r5)+n5,B ;2 find X+∂²X (L)     save ∂X (R)     preload x0 (R)
                macr    +x1,x0,B x:(r3)+n3,x0 A,y0          ;2 find X+∂²X (R)     prefetch ∂ (L)  save X+∂²X (L)

                mac     +x0,y0,A B,x0           y:(r5)+,y1  ;2 find X+∂²X+∂(X+∂²X) (L), save X+∂²X (R), prefetch width (L)

                mac     +x1,x0,B x:(r1)+n1,x0 y:(r6)+,y0    ;2 find X+∂²X+∂(X+∂²X) (R), prefetch e1 (L), Hns(1)
                asr      A                                  ;2 convert ES o/p @ 1/2 scale to NS i/p @ 1/4 scale
                asr      B                                  ;2 convert ES o/p @ 1/2 scale to NS i/p @ 1/4 scale
                ori      #$08,mr                            ;2 set up one asl on accumulator exits
                mac     +y0,x0,A x:(r2)+n2,x0 y1,y:(r5)+n5  ;2 1st filter tap (L),      "       (R), store width (L) over x0 (L)
                mac     +y0,x0,B x:(r1)+n1,x0 y:(r6)+,y0    ;2       "          (R), prefetch e6 (L), Hns(6)
                mac     +y0,x0,A x:(r2)+n2,x0 y:(r5)+n5,y1  ;2 2nd             (L),      "       (R), dummy read for postinc
                mac     +y0,x0,B x:(r1)+n1,x0 y:(r6)+,y0    ;2       "          (R), prefetch e4 (L), Hns(4)
                mac     +y0,x0,A x:(r2)+n2,x0 y:(r5)+,y1    ;2 3rd               (L),      "       (R), prefetch x1 (R)
                mac     +y0,x0,B x:(r1)+n1,x0 y:(r6)+,y0    ;2       "          (R), prefetch e2 (L), Hns(2)
                mac     +y0,x0,A x:(r2)+n2,x0 y1,y:(r5)+    ;2 4th               (L),      "       (R), store x1 (R)
                mac     +y0,x0,B x:(r1),x0      y:(r6)+,y0  ;2       "          (R), prefetch e7 (L) [no postinc], Hns(7)
                macr    +y0,x0,A x:(r2),x0      y:(r7)+,y1  ;2 last            (L),      "       (R) [     "     ], prefetch o/p scale
                macr    +y0,x0,B A,x0           y:(r7),A    ;2       "          (R), save sum2 (L), preload quantiser mask (L)
                and      x0,A    B,x1           y:(r7)-,B   ;2 find error (L), save sum2 (R), preload quantiser mask (R)
                and      x1,B    A,x:(r1)+      y:(r5)+,A   ;2 find error (R), store error (L) [postinc!], preload width LHS (L)
                mac     +x0,y1,A B,x:(r2)+      y:(r5)+,B   ;2 scale width THS (L), store error (R) [  "  ], preload width LHS (R)
                andi     #$f7,mr                            ;2 clear scaling
                mac     +y1,x1,B A,x:(r4)+      y:(r3)+,A   ;2 scale width THS (R), store width (L), preload input offset (R)
                tfr      A,B     B,x:(r4)-      y:(r3)+n3,y0 ;2 copy input offset (L), store width (R), prefetch input scale (B)
                m ve             A,x0                       ;2                      prefetch delay scalar

                jmp     loop 1                          ;4 re-enter wait-loop No.1 for next sample pair

                end                                     ;0

; end  f pr gram

; Phase l  ked l  p and Baud rate c ntrol calculations
  PWM  l ck 6   21 MHz, therefore, for 42 MHz operation, phase locked loop multiplier should be set to two (pctl=6e0001).
  Baud rate  f input = 10.50 MHz : 24 bits word, 2 channels separately, with Fs=437.5 KHz (Bck = 24 * 437.5 KHz = 10.50 MHz).
; wrt  system cl ck = 42 10.5 MHz. ie: baud rate = 1 4 of internal clock (for which brc = 0101 ... 1/2,1/2).   nb: WCk=220 KHz!
, Baud rate  f output: 10.5 MHz (defined by brc), only 10 LSBs used : delay, width, alternately, magnitude data assumed.
  Notes
, Register Usage
; R [       X $ffe2      N0 =         M0 =         Rx port
; R1        X $   0      N1 = a       M1 = d       left error
  R2  ]     X $  10      N2 = a       M2 = d       right error
, R3  ]     X $   2 ____ N3 = 2 ____  M3 = 5 ____  program RAM: logical & scalar constants (X&Y-data)
, R4  ]     X $ffe5      N4 =         M4 =         Tx port
  R         Y $          N5 = 2       M5 = 5       program RAM: i p buffer and L&R intermediate values
  R6  ]     Y·$  10      N6 =         M6 = 4       filter coeffs
  R   ]     Y·$   2      N7   2       M7   3       logical & scalar constants (Y-data only)
,  rganisati n of c nstants
, X R3  ]   47d27d ( (1 ε) R.  )     Y·R7[0] = 000100 (  output scale  )
  X R3 1]   7fffff ( full scale )     Y:R7[1] = 007fff ( Q(z) AND mask  )
;         pRAM                      Y:R7[2] = 00000a (  delay offset  )
,         pRAM                      Y:R7[3] = 000123 (  width offset  )
,                                                                       Y:R3[4] = 3d70a4 (   i p scale.   )
                                                                        Y:R3[5] = 400000 (  input offset  )
,  rganisati n of program RAM
,                   S1                                          -               S2
                      Y:R5[0]  previous i p (L)                 -                 Y:R5[0]  width LHS (R)
,                     Y:R5[1]  current i p (L)                  -                 Y:R5[1]  current i p (L)
  X R3[2]   1 x  (L)  Y:R5[2]  current i p (R)                  - X:R3[2] = ∂ (L)  Y:R5[2]  previous i p (R)
; X R3[3]   1 x0 (R)  Y:R5[3]  previous i p (R)   ∂ (L)        -                 Y:R5[3]  current i/p (R)
                      Y:R5[4]  width LHS (L)                    -                 Y:R5[4]  width LHS (L)
;                     Y:R5[5]  width LHS (R)                    -                 Y:R5[5]  previous i/p (L)
, nb  After S1 ES, Y:R5[1] is copied over Y:R5[5] (via Y:R5[3]) and R5 is then decremented to point to [4],
      after S1 NS, Y:R5[4] is written as shown but Y:R5[5] is written to Y:R5[0] to complete a swap of [5] & [0] as implied.
,     After S2 ES, Y:R5[4] is copied over Y:R[5] and then Y:R5[3] is copied over Y:R5[4] to restore indices as during S1.
;     R5 is decremented after S1 (so that Y:R5[2,4] are ready for S2) and incremented after S2 to restore initial conditions.
;  rganisation of loop error values (coeff order: 1,6,4,2,7,inc before next sample)
;         X:R1[0]  lde1              X:R2[0]  rde1
;         X:R1[1]      lwe2          X:R2[1]      rwe2
;         X:R1[2]  lde2              X:R2[2]  rde2
;         X R1[3]      lwe3          X:R2[3]      rwe3
;         X:R1[4]  lde3              X:R2[4]  rde3
;         X:R1[5]      lwe4          X:R2[5]      rwe4
;         X:R1[6]  lde4              X:R2[6]  rde4
;         X:R1[7]      lwe5          X:R2[7]      rwe5
;         X:R1[8]  lde5              X:R2[8]  rde5
;         X:R1[9]      lwe6          X:R2[9]      rwe6
;         X:R1[a]  lde6              X:R2[a]  rde6
;         X:R1[b]      lwe7          X:R2[b]      rwe7
;         X:R1[c]  lde7              X:R2[c]  rde7
;         X:R1[d]      lwe1          X:R2[d]      rwe1
```

Bibliography

(nb. Publications are referred to by the first three letters of the primary author's surname followed by the year of publication. Where this would lead to ambiguity, the papers are listed chronologically and a letter is appended to the reference to reflect this. References are listed here in alphabetical order as far as possible).

ABR65 "Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables",
edited by M. Abramowitz, I.A. Stegun,
published by Dover Publications, Inc., New York, 1965.

ADA92 "VLSI Architectures for Asynchronous Sample-Rate Conversion",
R. Adams, T. Kwan,
presented at the 93rd. A.E.S. Convention, October 1992, San Francisco, preprint No. 3355.

ADI93 "Stereo Asynchronous Sample Rate Converters" (preliminary information data sheet),
Analog Devices Inc.,
February 1993.

ATK89 "An Introduction to Numerical Analysis",
K.E. Atkinson,
published by Wiley, New York, 1989.

ATT83 "Design Parameters Important for the Optimisation of Very High Fidelity PWM (Class D)
B.E. Attwood, Audio Amplifiers",
Journal of the Audio Engineering Society, Volume 31, Number 11, November 1983.

BEL74 "Interpolation, Extrapolation, and Reduction of Computation Speed in Digital Filters",
M.B. Bellanger, J.L. Daguet, G.P. Lepagnol,
I.E.E.E. Trans.Acoust.Speech Signal Process,Vol.ASSP-22, No.4, pp.231-235, Aug.1974.

BER69 "A Radix-Eight Fast Fourier Transform Subroutine for Real-Valued Series",
G.D. Bergland,
I.E.E.E. Trans. on Audio and Electroacoustics, Vol. AU-17, No.2, pp. 138-144, June 1969.

BIN67 "Modern Techniques of Power Spectrum Estimation",
C. Bingham, M.D. Godfrey, J.W. Tukey,
I.E.E.E. Trans. on Audio and Electroacoustics, Vol. AU-15, No. 2, pp. 56-66, June 1967.

BLA53 "Modulation Theory",
H.S. Black,
published by D. Van Nostrand Co, Inc., New Jersey, 1953.

BLA87 "Fast Algorithms for Digital Signal Processing",
R.E. Blahut,
published by Addison Wesley, Inc., Reading, MA, 1985.

BOI81 "A New Procedure for the Design of High Order Minimum Phase FIR Digital or CCD
R. Boite, H. Leich, Filters",
Signal Processing, Vol. 3, pp. 101-108, 1981 (North Holland Publishing Company).

BOW91 "Performance & Design of a Noise Shaping PWM DAC",
R.G. Bowman, R.E. Hiorns & M.B. Sandler,
presented at the I.E.E. Colloquium for Digital Audio, London, May 1991, digest No.107.

BUL88 "Primitive Operator Digital Filter Synthesis using a Shift Biased Algorithm",
D.R. Bull, D.H. Horrocks,
I.E.E.E. Intl. Symp. on Circuits & Systems, Helsinki, Finland, pp.1529-1532, June 1988.

CAN85 "A Use of Double Integration in Sigma Delta Modulation",
J.C. Candy,
I.E.E.E. transactions on communications,vol. COM - 33, pp. 249-258, March 1985.

CAN86 "Decimation for Sigma Delta Modulation",
J.C. Candy,
I.E.E.E. transactions on communications,vol. COM - 34, pp. 72-76, January 1986.

CAR87 "An Oversampling Analogue to Digital Converter Topology for High Resolution Signal
L.R. Carley, Acquisition Systems",
I.E.E.E. transactions on circuits & systems, vol. CAS - 34, no 1, January 1987.

CHA90 "A Higher Order Topology for Interpolative Modulators for Oversampling A/D Converters",
K.C.-H. Chao et. al..,
I.E.E.E transactions on circuits and systems, CAS - 37, pp. 309-318, March 1990.

CHE92 "A Novel Fast Filtering Algorithm and its Implementation.",
Sau-Gee Chen, Ren-Jer Tsai,
Signal Processing VI, pp.945-948, proceedings of EUSIPCO '92, Brussels, October 1992.

CLA93 "V.H.D.L. Implemented Integer Noise Shapers with Floating Point Accuracy",
I.G. Clark, R.E. Hiorns & A.C. Davies,
presented at the 94 th. Audio Engineering Society Conference, Berlin, March 1993.

COR84 "Simultaneous Design in Both Magnitude and Group Delay of IIR and FIR Filters Based on Multiple Criterion Optimisation.",
G. Cortelazzo, M.R. Lightner,
I.E.E.E. Trans. Acoust.Speech Signal Process.,Vol.ASSP-32, No.5, pp.949-967,Oct.1984.

COR87 "Minimising Multimodal Functions of Continuous Variables with the Simulated Annealing Algorithm",
A. Corona, M. Marchesi, C. Martini, S. Ridella,
ACM Transactions on Mathematical Software, vol.13, pp. 262-230, 1987.

CRA92 'Towards the 24-bit DAC : novel Noise-Shaping Topologies incorporating Correction for the Non-linearity in a PWM Output Stage.",
P.G. Craven,
Consultant's report to B & W Loudspeakers Ltd., August 1992,
and subsequently : Journal of the Audio Engineering Society, Vol. 41, No. 5, May 1993.

CRO83 "Multirate Digital Signal Processing",
R.E. Crochiere, L.R. Rabiner,
published by Prentice Hall, Englewood Cliffs, New Jersey, 1983.

DEC72 "Synthesis of Recursive Digital Filters Using the Minimum - ρ Error Criterion.",
A.G. Deczky,
I.E.E.E. Trans. on Audio and Electroacoustics, Vol. AU-20, No.4, pp.257-263, Oct.1972.

DEI92 "HV1000 Pulser" (advance information data sheet), September 1992,
Directed Energy Inc.,
2301 Research Boulevard, Site 101, Fort Collins, Colorado 80526, U.S.A.

FAR93 "Development of a Commercial Digital Amplifier"
H.D. Farrar, R.E. Hiorns
presented at the I.E.E. Colloquium for Audio DSP, London, Nov. '93, digest No.1993/219.

FDI86 "Active Filter Products' Design and Selection Guide.",
Frequency Devices, Inc.,
25 Locust Street, Haverhill, Massachusetts 01832, USA, 1986.

FLE63 "A Rapidly Convergent Descent Method For Minimisation.",
R. Fletcher, M.J.D. Powell,
Computer Journal, Volume 6, No.2, pp. 163-168, 1963.

GBL91 "1991 GaAs IC Data Book and Designers Guide.", August 1990,
Gigabit Logic Inc.,
1908 Oak Terrace Lane, Newbury Park, California 91320, U.S.A.

GER89 "Optimal Noise Shaping and Dither of Digital Signals",
M. Gerzon, P.G. Craven,
presented at the 87 th. A.E.S. Convention, September 1989, New York, preprint No. 2822.

GHA91 "Implementation Of Recursive Filters Using Highly Quantised Periodically Time-varying
S. Ghanekar, S. Tantaratana, L.E. Franks, Coefficients",
I.E.E.E. Transactions, ref. CH2977-7/91/0000-1625, 1991.

GIN91 "A New Variable Fractional Sample Delay Filter with Non-linear Interpolation",
Ging-Shing Liu, Che-Ho Wei,
I.E.E.E., Ref. CH3006-4/91/0000-2451, 1991.

GOL69 "Digital Processing of Signals",
B. Gold, C. Rader,
published by the McGraw-Hill Book Company, New York, 1969.

GOL91 "Pseudo-natural pulse width modulation for high accuracy digital-to-analogue conversion",
J.M. Goldberg, M.B. Sandler,
Electronics Letters, August 1991.

GOL92 "Signal Processing for High Resolution Pulse Width Modulation Based Digital-to-Analogue
J.M. Goldberg, Conversion",
PhD. Thesis, December 1992.

GOO77 "Nine Digital Filters for Decimation and Interpolation",
D.J. Goodman, M.J. Carey,
I.E.E.E. Trans. Acoust.Speech Signal Process.,Vol.ASSP-25,No.2,pp.121-126,April 1977.

GRA71 "An Always Convergent Minimisation Technique for the Solution of Polynomial
J.A. Grant, G.D. Hitchins, Equations."
Journal of Inst. Math. Appl., Volume 8, pages 122-129, 1971.

HAU90 "Overview of Oversampling A/D Conversion",
M.W. Hauser,
Presented at the 89 th. A.E.S. Convention, Los Angeles, September 1990.

HAW92 "Dynamic Model Based Linearisation of Quantised Pulse Width Modulation for
Applications in Digital to Analogue Conversion and Digital Power Amplifier Systems",
M.O.J. Hawksford,
Journal of the Audio Engineering Society, Volume 40, No. 4, pp. 235-252, April 1992.

HER70 "On the Design of Non-Recursive Digital Filters with Linear Phase.",
O. Herrmann, H.W. Schuessler
Electronic Letters, Volume 6, Number 11, pages 328 to 329, 1970.

HIO88 "PWM Class D Amplifier",
R.E. Hiorns,
Thesis, submitted in part fulfilment of the Bachelor of Arts Degree requirements,
Cambridge University, April 1988

HIO89 "PWM Modulator for Class D Audio Amplification",
R.E. Hiorns,
Thesis, submitted in part fulfilment of the Master of Science Degree requirements,
University of London, Sept. 1989.

HIO90a "PWM Modulator for Class D Audio Amplification,
R.E. Hiorns, Additional Information and Further Work.",
King's College internal report to Dr. M.B. Sandler, March 1990.

HIO90b "Realising an All Digital Power Amplifier",
R.E. Hiorns, J.M. Goldberg & M.B. Sandler,
presented at the 89 th. A.E.S. Convention, Los Angeles, October 1990, preprint No. 2959.

HIO91a "Developments in Realising an All Digital Power Amplifier",
R.E. Hiorns, R.G. Bowman, J.M. Goldberg & M.B. Sandler,
presented at the A.E.S. 90 th. Convention, Paris, February 1991, preprint No. 3034.

HIO91b "Design Limitations for Digital Audio Power Amplification",
R.E. Hiorns, J.M. Goldberg & M.B. Sandler,
presented at the I.E.E. Colloquium for Digital Audio, London, May 1991, digest No.107.

HIO91c "A PWM DAC for Digital Audio Conversion : from Theory to Performance",
R.E. Hiorns, R.G. Bowman & M.B. Sandler,
presented at the I.E.E. International Conference on A/D & D/A, Swansea, September 1991.

HIO92 "Power Digital to Analogue Conversion using Pulse Width Modulation and Digital
R.E. Hiorns, M.B. Sandler, Signal Processing",
submitted to I.E.E. Proceedings-G, July 1992, paper No.CDS /92/1880,
published in revised form IEE Proceedings-G, Vol. 140, No. 5, October 1993.

HIO93 "A Modified Noise Shaper Structure for Digital PWM DACs",
R.E. Hiorns, A.C. Paul & M.B. Sandler
presented at the 95 th. Audio Engineering Society Conference, New York, September 1993.

HOF71 "A New Technique for the Design of Non-Recursive Digital Filters",
E. Hofstetter, A.V. Oppenheim, J. Siegel,
Proc. 5 th. Annual Princeton Conf. on Inform. Sci. Systems, pages 64 to 72, 1971.

HOO61 "Direct Search Solution of Numerical and Statistical Problems",
R. Hooke, T.A. Jeeves,
Journal of the Association of Computer Mach., Volume 8, No.4, pp. 212-229, April 1961.

HOR80 "The Art of Electronics",
P. Horowitz, W. Hill,
published by the Press Syndicate of the University of Cambridge, The Pitt Building,
Trumpington Street, Cambridge, CB2 1RP.

HOR91 "Design and Implementation of Sigma-Delta D/A Converters with Optimised Loop Filters",
U. Horbach, M. Lang,
I.E.E.E., Ref. CH 3006-4/91/0000, pp. 1633-1636, April 1991.

KNO89 "Bit-Level Systolic Architectures For High Performance IIR Filtering",
S.C. Knowles, J.G. McWhirter, R.F. Woods, J.V. McCanny,
Journal of VLSI Signal Processing, Volume 1, pp. 9-24, 1989.

LEI90a "Distortion Minimisation in Pulse Width Modulated Systems using a Digital Sampling
S.P. Leigh, P.H. Mellor, B.M.G. Cheetham, Process",
Electronics Letters, Volume 26, No. 16, pp. 1310,1311, 2 nd. August 1990.

LEI90b "The Implementation and Performance Enhancement of a Completely Digital Power
S.P. Leigh, P.H. Mellor, B.M.G. Cheetham, Amplifier",
Proceedings of the Institute of Acoustics, Volume 12, part 8, pp. 67-75, 1990.

LIM88 "Advanced Topics in Signal Processing",
J.S. Lim, A.V. Oppenheim,
published by Prentice Hall, Englewood Cliffs, New Jersey, 1988.

LIP89 "High Pass Dither",
S.P. Lipshitz, J.R. Vanderkooy,
presented at the A.E.S. 4 th. Regional Convention, June 1989, Tokyo, paper A5.

LIP91 "Quantisation and Dither : a Theoretical Survey",
S.P. Lipshitz, R.A. Wannamaker, J.R. Vanderkooy,
presented at the 91 st. A.E.S. Convention, October 1991, New York, preprint 3141.

MAL67 "Single Polarity Pulse Width Modulation Of the Third Kind",
V.V. Malanov,
Journal of Telecommunication and Radio Engineering, Volume 22, Part 2, pp. 67-70, 1967.

MAR70 "Theoretical Efficiencies of Class D Power Amplifiers",
J.D. Martin,
Proceedings of the I.E.E, Vol.117, No.6, June 1970, pp 1089 - 1090.

MAR79 "A Class of Infinite-Duration Impulse Response Digital Filters for Sample Rate Reduction"
H.G. Martinez, T.W. Parks,
I.E.E.E. Trans.Acous.Speech Signal Process.,Vol.ASSP-27, No.2, pp.154-162, Apr. 1979.

MAR92 "IIR Filter Design using Mixed Simulated Annealing-Deterministic Optimisation",
M.L. Marchesi
I.E.E.E. 0-7803-0593-0/92

MAT89 "A 17 Bit Oversampling D/A Conversion Technology Using Multistage Noise Shaping",
Y. Matsuya, et. al.. ,
I.E.E.E. Journal of Solid State Circuits, SC-24, August 1989, pp. 969-975.

MCC73a "A Unified Approach to the Design of Optimum FIR Linear Phase Digital Filters",
J.H. McClellan, T.W. Parks,
I.E.E.E. Trans. on Circuit Theory, Volume CT-20, pages 697 to 701, November 1973.

MCC73b "A Computer Program for Designing Optimum FIR Linear Phase Digital Filters",
J.H. McClellan, T.W. Parks, L.R. Rabiner,
I.E.E.E. Trans. on Audio and Electroacoustics, Vol. AU-21, No.6, pp.506-526, Dec. 1973.

MEL91 "Reduction of Spectral Distortion in Class D Amplifiers by an Enhanced Pulse Width
P.H. Mellor, S.P. Leigh, B.M.G. Cheetham, Modulation Process",
Proceedings of the I.E.E., part G, Volume 138, No. 4, pp. 441-448, August 1991.

MET53 "Equation of State calculations by Fast Computing Machines",
N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller,
Journal of Chemistry and Physics, Vol. 21, pp. 1087-1090, June 1953.

NEL64 "A Simplex Method For Function Minimisation",
J.A. Nelmer, R. Mead
The Computer Journal, vol.7, pp. 308-313, 1964.

NIE89 "Design of Linear-Phase Direct-Form FIR Digital Filters with Quantised Coefficients Using
J.J. Nielsen, Error Spectrum Shaping Techniques",
I.E.E.E. Trans. Acous., Speech, Signal Process., ref. 0096-3518/89/0700-1020, 1989.

NIC91 "An Investigation into the Multiple Criterion Optimisation Approach to IIR Digital Filter
L.J. Nicolson, B.M.G. Cheetham, Design",
I.E.E. Colloquium on Filter Design (E10), London, December 1991.

OPP75 "Digital Signal Processing",
A.V. Oppenheim, R.W. Schafer :
published by Prentice Hall, Englewood Cliffs, New Jersey 1975.

PAR72 "Chebyshev Approximation for Non-Recursive Digital Filters with Linear Phase.",
T.W. Parks, J.H. McClellan,
I.E.E.E. Trans. on Circuit Theory, Vol. CT-19, pp.189194, March 1972.

PAR92 "McGraw Hill Encyclopedia of Science and Technology",
Edited by S.P. Parker,
published by McGraw Hill, Inc., New York, N.Y., U.S.A., 7 th. edition, 1992.

PAU92a "Steps Towards Developing a High Resolution D/A Converter using Pulse Width
A.C. Paul, Modulation.",
MPhil. 'Transfer' Thesis, King's College, University of London, submitted June 1992.

PAU92b A.C. Paul, private communication, September 1992.

PED94 "All Digital Power Amplifier Based on Pulse Width Modulation",
M.S. Pedersen, M. Shajaan,
presented at the A.E.S. 96 th. Convention, Amsterdam, February 1994, preprint No. 3809.

RAB75a "Theory and Application of Digital Signal Processing.",
L.R. Rabiner, B. Gold,
Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

RAB75b "FIR Digital Filter Design Techniques Using Weighted Chebychev Approximation.",
L.R. Rabiner, J.H. McClellan, T.W. Parks,
Proceedings of the I.E.E.E., Volume 63, Number 4, pages 595 to 610, April 1975.

REM57 "General Computational Methods of Tchebycheff Approximation.",
E. Ya. Remez,
Kiev, (Atomic Energy Translation 4491), pp. 1-85, 1957.

RIB91 "A Comparison of Modulator Networks for High-Order Over-sampled ΣΔ Analogue to
D.B. Ribner, Digital Converters",
I.E.E.E. Transactions on Circuits & Systems, February 1991.

RIC82 "Application of Deczky's Program for Recursive Filter Design to the Design of Recursive
M.A. Richards, Decimators",
I.E.E.E. Trans.Acous.Speech Signal Process.,Vol.ASSP-30, No.5, pp.811-814, Oct. 1982.

RIL91 "High Decimation Digital Filters",
C. Riley, D. Chester, A. Razavi, F. Taylor, W. Ricker,
I.E.E.E. Ref. CH2977-7/91/0000-1613, published 1991.

ROB62 "Picture Coding Using Pseudo-Random Noise",
L.G. Roberts,
I.R.E. Transactions on Information Theory, Vol. IT - 8, pp. 145-154, February 1962.

ROW65 "Signals and Noise in Communication Systems",
H.E. Rowe,
D. Van Nostrand Co. Inc., Princeton, New Jersey, 1965.

SAN83 "Digital Power Amplifier Design",
M.B. Sandler,
PhD. Dissertation, University of Essex, Colchester, U.K., 1983.

SAN91a "Ultra Low Distortion Digital Power Amplification",
M.B. Sandler, J.M. Goldberg, R.E. Hiorns, R.G. Bowman, M. Watson & P. Ziman
presented at the A.E.S. 91 st. Convention, New York, October 1991, preprint No. 3115.

SAN91b "Digital To Analogue Conversion using PWM",
M.B. Sandler, J.M. Goldberg & R.E. Hiorns,
UK Patent Application No. 9027503.3, December 1991.

SAN91c Private communication with Dr. M.B. Sandler, May 1991.

SAN92 "Digital To Analogue Conversion using PWM"
M.B. Sandler, J.M. Goldberg & R.E. Hiorns,
PCT. Patent Application No. PCT./GB91/02279, August 1992.

SAR90 "Multiplier Free Decimator Algorithms for Super-resolution Oversampled Converters
T. Saramaki, T. Karema, T. Ritoniemi, H. Tenhunen,
I.E.E.E. Symposium on Circuits & Systems, New Orleans, May 1990, vol. 4, pp. 3275-8.

SCH89 "Numerical Analysis : A Comprehensive Introduction",
H.R. Schwarz,
published by Wiley, Chichester 1989.

SED82 "Micro-Electronic Circuits",
A.S. Sedra, K.C. Smith,
published by C.B.S. College Publishing, 383 Madison Avenue, New York, U.S.A., 1982.

SIL89 "Sub-nanosecond Switching with DMOS FETs",
Application note LPD11 (Ed Oxner, et al., Low Power Discretes Data Book, pp 9/100-106)
Siliconix Inc., 2201 Laurelwood Rd, Santa Clara, CA 95054, 1989.

SRI77 "A Necessary and Sufficient Condition for Quantisation Errors to be Uniform and White",
A.B. Sripad, D.L. Snyder,
I.E.E.E. Trans.Acous.Speech, Signal Proc., ASSP -25, No.5, pp. 442-448, October 1977.

TEW78 "Oversampled, Linear Predictive and Noise Shaping Coders of Order N>1",
S.K. Tewkesbury, R.W. Hallock,
I.E.E.E. Trans. (Circuits and Systems), July 1978.

VAN80 "Feedforward Error Correction in Power Amplifiers",
J.R. Vanderkooy, S.P. Lipshitz,
Journal of the A.E.S., Volume 28, No. 1/2, pp. 2-16, January / February 1980.

VAN89 "Digital Dither : Signal Processing with Resolution Far Below the Least Significant Bit.",
J.R. Vanderkooy, S.P. Lipshitz,
presented at the 7 th. A.E.S. International Conference, May 1989, Toronto.

VET88 "Running FIR and IIR Filtering Using Multirate Filter Banks",
M. Vetterli,
I.E.E.E. Trans., Acous. Speech, Signal Process., Vol. ASSP-36, pp. 730-738, May 1988.

WAL75 "Current Dumping Audio Amplifier",
P.J. Walker, M.P. Albinson,
presented to the 50 th. Convention of the A.E.S., March 1975, London.

WEG24 "The Auditory Masking of One Sound by Another and its Probable Relation to the
R.L. Wegel, C.E. Lane, Dynamics of The Inner Ear",
Physics Review, No. 23, pages 266-285, 1924.

WHI84 "Concepts of Scale in Simulated Annealing",
S.R. White,
Proceedings of the I.E.E.E., ICCD '84, New York, pp. 646-651, 1984.

WON93 "Simulation of Clock Jittering in Pulse Width Modulated, Digital to Analogue Power
D. Wong, Converters",
Thesis, submitted in part fulfilment of the Master of Engineering Degree requirements,
University of London, 1993.

YOS77 "Fundamentals of Hearing : An Introduction.",
W.A. Yost, D.W. Nielsen,
published by Holt, Rinehart & Winston, 1977.

Addendum

HAR78 "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform.",
F.J. Harris,
Proceedings of the I.E.E.E., volume 66, number 1, January 1978.

NUT81 "Some Windows with Very Good Sidelobe Behaviour",
A.H. Nuttall,
IEEE Transactions, Acous., Speech & Signal Proc., Vol. ASSP-29, No.1, February 1981.

List of Abbreviations

(note : Acronyms are listed numerically before alphabetically, with special characters (eg. Greek) at the end).

2SCPWM	two sample, consecutive, PWM (uniformly sampled only)
74AC???	advanced CMOS (IC logic family)
74HC???	high speed CMOS (IC logic family)
74F???	high speed TTL (IC logic family)
74LS???	low power Schottky TTL (IC logic family)
74S???	'industry standard' TTL (IC logic family)
AB	class AB : mixed class A, class B operation
AC	alternating current
ACF	auto-correlation function
AD	two level (PWM output type)
ADC	analogue to digital converter
A/D	analogue to digital (converter / conversion)
AM->PM	amplitude modulation to phase modulation (conversion)
AOAPWM	alternating odd-valued asymmetric PWM (uniformly sampled only)
APF	all pass filter
ASIC	application specific integrated circuit
BD	three level (PWM output type)
BPF	band pass filter
BJT	bipolar junction transistor
CCD	charge coupled device
CD	compact disc
CD4???BE	CMOS 4000B series (logic family)
CMOS	complimentary metal oxide substrate (IC technology)
D/A	digital to analogue (converter / conversion)
DAB	digital audio broadcast
DAC	digital to analogue converter
DAT	digital audio tape
dB	decibels
dBFS	decibels with respect to a full scale signal
DC	direct current (in a DSP context, zero frequency)
DCC	digital compact cassette
DFT	discrete Fourier transform
DLE	differential linearity error
DMOS	D-geometry MOSFET (double diffused power MOSFET)
DPWM	digital pulse width modulator / modulation
DSB	double side band (phase modulation)
DSPWM	double sided PWM (naturally sampled only)
DTFT	discrete time Fourier transform

DTL	diode-transistor logic
ECL	emitter coupled logic (10K/100K IC logic families)
EMC	electromagnetic compliance (to European Community requirements)
EMI	electromagnetic interference
EOC	'end of count' (a logical signal used in counter such as in PWMs)
ESPWM	enhanced sampled pulse width modulation
ESR	equivalent series resistance
ESS	error spectral shaper / shaping (noise shaping)
ES2	European Silicon Structures (ASIC design & foundry)
EXOR	exclusive OR (logic function)
FACT	Fairchild advanced CMOS technology (see 74AC)
FAST	see 74F
F_c	carrier frequency (=PRR)
FET	field effect transistor
FFT	fast Fourier transform
FIR	finite impulse response
FM	frequency modulation
FREDFET	fast recovery epitaxial diode field effect transistor
FPGA	field programmable gate array
F_s	sampling frequency
FT	Fourier transform
F/V	frequency to voltage (converter / conversion)
GaAsFET	gallium arsenide field effect transistor
GAL	gate array logic
HEXFET	hexagonal geometry, MOSFET array FET (power MOSFET)
HPF	high pass filter
IC	integrated circuit
IDFFT	inverse discrete fast Fourier transform
IDFT	inverse discrete Fourier transform
IDTFT	inverse discrete time Fourier transform
IIR	infinite impulse response
IMD	inter-modulation distortion
I/O	input & output
LAPWM	lagging asymmetric PWM (uniformly sampled only)
LEPWM	leading edge PWM
LPWM	(partially) linearised pulse width modulation (from [PED94])
LPF	low pass filter
LSB	least significant bit
LTI	linear, time-invariant (process / system)
MBF	multi-band filter
MLSR	maximal length shift register

MLSC	maximal length sequence counter
MMIC	monolithic microwave intergrated circuit
MOSFET	metal oxide substrate field effect transistor
MPEG	moving pictures expert group (data compression standards)
MP-NS	minimum phase noise shaper
MSB	most significant bit
N/C	noise power per carrier power (in dBs)
NLSB	next least significant bit
NMSB	next most significant bit
NPN	n-doped, p-doped, n-doped BJT
NRZ	non-return to zero (pulse type)
NTF	noise transfer function
NEXOR	NOT exclusive OR (logical function)
ONS	oversampled noise shaper / shaping
OS	oversampling (ratio...eg. 4x, 8x, etc.)
PAL	programmable array logic
PAM	pulse amplitude modulator / modulation
PCB	printed circuit board
PCM	pulse code modulation
PDM	† pulse density modulator / modulation
PDF	probability density function
PEM	pulse edge modulation (a commercial name for low bit PWM)
PID	proportional-integral-differential (control strategy)
PLL	phase locked loop
PM	phase modulation
PNP	p-doped, n-doped, p-doped BJT
PNPWM	pseudo-naturally sampled PWM
PPM	pulse position modulator / modulation
PRNG	pseudo-random noise generator
PRR	pulse repetition rate ($=F_c$)
PSD	power spectral density
PSU	power supply unit
PWM	pulse width (duration †) modulator / modulation
RAM	random access memory
RMS	root-mean-square
ROM	read only memory
RTL	resistor-transistor logic
SBC	sub-band coding
SMPS	switched mode power supply
SNR	signal to noise ratio

† PDM is used in some texts to refer to pulse duration modulation, ie. PWM (eg. BLA53).

SPDIF	Sony / Phillips' digital interface format
STF	signal transfer function
SYMPWM	symmetric PWM (uniformly sampled only)
TEPWM	trailing edge PWM
THD	total harmonic distortion
TTL	transistor-transistor logic
t_{phl}	propagation delay (high to low going output)
t_{plh}	propagation delay (low to high going output)
TVI F	time-variant, linear filtering
UEE	uncorrelated edge error
VCO	voltage controlled oscillator
VMOS	V-geometry MOSFET (power MOSFET)
ZI-NS	zero interleaved noise shaper
$\Sigma\Delta$	sigma delta (modulator / modulation)

Colophon

This document was produced on an
Apple Macintosh SE,
running system 6.0 'MultiFinder',
using
Claris MacWrite II,
Claris MacDraw II
&
Aldus PageMaker 4.0
and stored in compressed form using
Compact Pro.

Imported graphics were embedded in
Adode 1.0, Postscript
from
Simulate (A.C. Paul, KCL)
System One (Audio Precision Inc.),
Cadstar (Racal Redac Ltd.)
&
SOLO (European Silicon Structures)
after pre-processing with the Unix utilities, *'Awk' & 'Sed'*
or
scanned using an
Apple 300 dpi. Scanner
and embedded in
compressed TIFF.

The final copy was rendered at 600 dpi.
using a
Hewlett Packard *Laserjet 4M.*
on
Mellotex (105 gm^{-2})